Dock Door Case Study

in collaboration with Mojix

Marlize DE VILLIERS
Ayoub YOUSSOUFI

March 29, 2022

Contents

List of Tables

List of Figures

Acronyms

RFID	Radio-Frequency Identification
IoT	Internet of Things
RSSI	Received Signal Strength Indication
RC	Read count of the RSSI
MLP	Multi-layer perceptron
EPC	Electronic Product Code
kNN	k-Nearest Neighbours
SVM	Support Vector Machine
SMOTE	Synthetic Minority Oversampling TEchnique

1 Introduction

Industry 4.0 and the technology that comes with it has gained popularity over the last few years. One innovative application of Industry 4.0 is inventory management, and specifically, the automation thereof. In short, inventory management is concerned with monitoring stock as it arrives, moves around, and leaves the facility. One way in which Industry 4.0 is playing a role in inventory management is by automating inventory management using Radio-Frequency Identification (RFID) technology.

Automated inventory management has recently became a point of interest for many businesses, since automated inventory management can save time, make people's jobs easier, and reduce the number of human errors made. This all has a positive impact on the business's overall productivity and efficiency. These aspects are all important for businesses who wish to remain competitive — especially in the time of Industry 4.0.

1.1 Company background

Mojix is a leading software company that focuses their building solutions for their customers — whether they are consumers, retailers, or industry enterprises — using data gathered by various Internet of Things (IoT) technologies. One of Mojix's focuses is digitizing and automating supply chains. They do this in various ways. This case study focuses on automated inventory management. The way that Mojix has decided to approach this problem is using RFID technology.

RFID technology enables Mojix to automate an inventory system by allowing communication between RFID tags and sensors. Data are transmitted between the tags and the sensors, automating the process of monitoring stock in a facility without the need for humans to interfere or assist [4]. Mojix already has item chain management solutions — shown in Figure 1 — that help their clients manage the life-cycle of the products in terms of cost, time and quality of the operations to deliver unique end-to-end costumer experience.

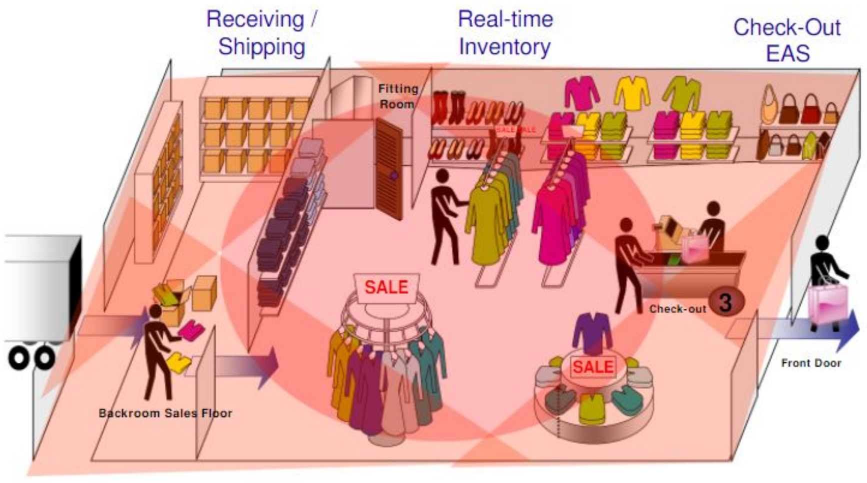


Figure 1: An example of an item management solution provided to clothing store [8]

1.2 Problem statement

When this case study was started, the Dock Door project was very recently started at Mojix. As a result, this case study contributed to refining the problem definition for Mojix since as work was done, new problems and ideas came up and were included into the scope. The case study, therefore, aided in developing the project. The final problem statement, therefore is: *Find a way to automate the inventory management system using machine learning without compromising on performance.*

From this problem statement, three main tasks were identified:

1. Accurately distinguishing between moving and static RFID tags.
2. Accurately determine if moving tags are coming into the facility or leaving the facility.
3. Accurately determine the time that tags entered or left the facility.

This case study focuses mainly on these three tasks.

The rest of the report contains the *Literature review* which discusses some technical concepts and previous work, followed by a description of the data used in the *Dataset* section. The algorithms used in the case study are briefly explained in the *Algorithms* section. Then, in the *Results*, each problem identified in the problem statement is described and the machine learning algorithms' performances are discussed. Lastly, the findings of this case study and our advice to Mojix regarding future work are contained in the *Conclusion and recommendations*.

2 Literature review

The Industry 4.0 concept opens opportunities to digitize industrial machines, processes, and assets; providing new insights and efficiently counteracting various challenges. RFID is a key Industry 4.0 enabling technology, and has been widely used throughout manufacturing and supply chain areas such as logistics, clothing industry, library management, warehouse management and so on, and object localization has always been a hot topic in RFID research. In this section some concepts around RFID will be discussed, along with the use of RFID in inventory management, and finally, previous work on the same topic will be reviewed.

2.1 Radio-frequency identification

RFID technology is systematically replacing barcode identification. It simplifies and eliminates routine work of employees by manually detecting and identifying RFID tags using antennas, by removing the need to manually locate and scan barcodes to identify products. What makes RFID especially useful, is that a direct line of sight to the tagged item is not required for the antennas to detect the tags. Certain RFID technology can work over distances greater 20m [2]. All of this is why RFID is particularly suited for inventory management.

An RFID system comprises the following:

- Tags
- Readers
- Antennas

The tags are attached to the products of interest, and consist of two parts, an antenna that can send or receive signals, and a chip which stores the information about the tag — or its identification. Readers send and receive signals to and from the tags. Antennas essentially convert the signals from the readers into radio-frequency waves so that tags can be detected[2].

In short, the readers send signals, which are converted into radio-frequency waves by the antennas. When the radio-frequency waves reach a tag, the wave is reflected back to the antenna on the same path. The received radio-frequency wave is then converted back into a signal to enable the reader to receive the signal. This process enables us to determine the distance of the tag using the RSSI. The RSSI gives an indication of how strong the received signal is, which can then be used to determine the distance that the signal has travelled.

A main problem with the RSSI (and of RFID technology) is that the radio-frequency waves can "echo" and bounce off of metal surfaces. This becomes an even bigger problem in a closed environment, such as a warehouse. Unfortunately, the RSSI is the only measure available to determine the location of a tag relative to the readers, and if a more accurate location is required, it might be necessary to introduce additional technology [6]. Similar to this, the readers have no way to determine if a tag is moving or static, since the RSSI is a single value received at a time.

2.2 Problems when using radio-frequency identification for inventory management

In an inventory management system, an RFID system can monitor tagged products moving through an entrance or exit of a warehouse. However, RFID cannot distinguish signals received from tagged products that are actually moving through the gate from the signal that enter the reading range accidentally because of interference. This gives rise to a common problem in an RFID inventory management system — distinguishing true positives from false positives (i.e.: static tags are accidentally detected as moving tags are false positives, and are not of interest to the business process at the time that they are static.) Tags that move through the entrance or exit are called true positives.

In an attempt to minimise this problem, Mojix has a dedicated "in zone" and a dedicated "out zone", along with an "in antenna" aimed at the "in zone", and an "out antenna" aimed at the "out zone". The setup is shown in Figure 2. Though this does decrease false positives a little bit, it does not eliminate them completely. It also gives rise to an additional definition of false positives and false negatives: tags in a specific zone being detected by the other zone's antenna. This happens because of the echoing and interference discussed in the previous subsection.

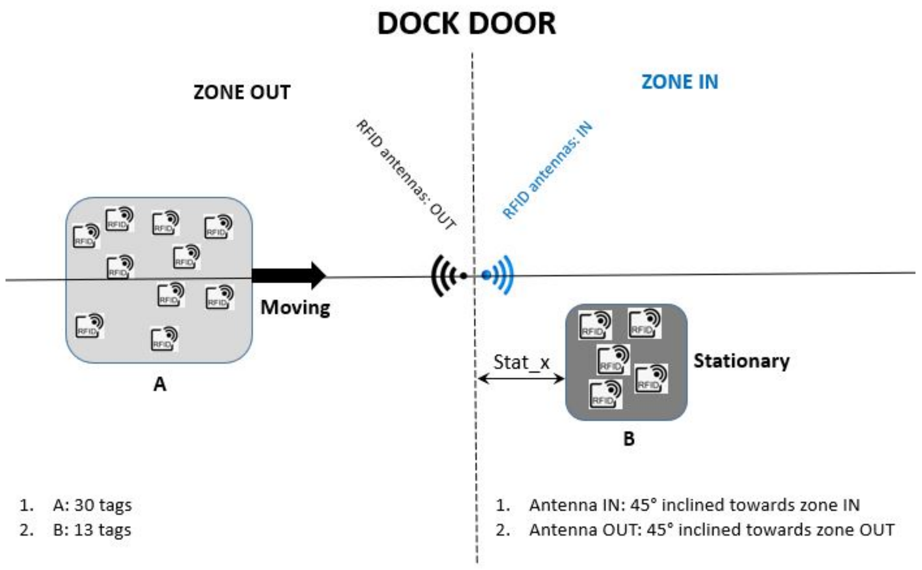


Figure 2: RFID system setup at the dock door in the warehouse [4].

2.3 Using machine learning to solve the problems posed by radio-frequency identification

Using machine learning for inventory management with RFID is not a new concept, but the literature is limited. Nonetheless, some insightful work has been done. The work that was done for this case study was mostly inspired by the first paper mentioned in this subsection.

Alfian et al. [1] used machine learning combined with RFID to improve the traceability of items in a supply chain. In their study, it was important to be able to determine the direction of movement of the RFID tags — since they wanted to be able to accurately track and trace items in the warehouses in the supply chain. The problem they tried to solve is very similar to the problem Mojix is trying to solve.

Alfian et al. [1] approached the problem by gathering data of various tag movements, namely:

1. Move from inside to outside.
2. Move from outside to inside.
3. Move around inside or move around outside.
4. Static inside or static outside.

The data that they gathered was the RSSI of the tags and the time the RSSI was received. They gathered the data for the tags being moved at various speeds, and the movement of tags going through the gate was always from 5m inside the one zone to 5m inside the other zone. After the data were gathered, they labeled the data appropriately according to the tags movements. The data thus contained, per run (a run is one completed movement of the tags), the following:

1. The timestamp.
2. The tag's identification.
3. The RSSI.
4. The antenna that received the RSSI.
5. The movement class.

Figure 3 shows the RSSI of the tags during various movements. Sub-figure a shows tags moving from outside to inside, and subfigure b shows tags moving from inside to outside. Sub-figure c shows tags moving around outside, and static tags inside, and sub-figure d shows tags moving inside close to the gate and then away again — without actually crossing the gate. We can see as the tags move through (or close to) the gate, the RSSI increases and they are first detected by one antenna, and as they approach, the RSSI for the first antenna decreases and they start to be detected by the second antenna with an increasing RSSI. If they move through, eventually, they are only detected by the second antenna. Static tags have a constant RSSI, and tags that move

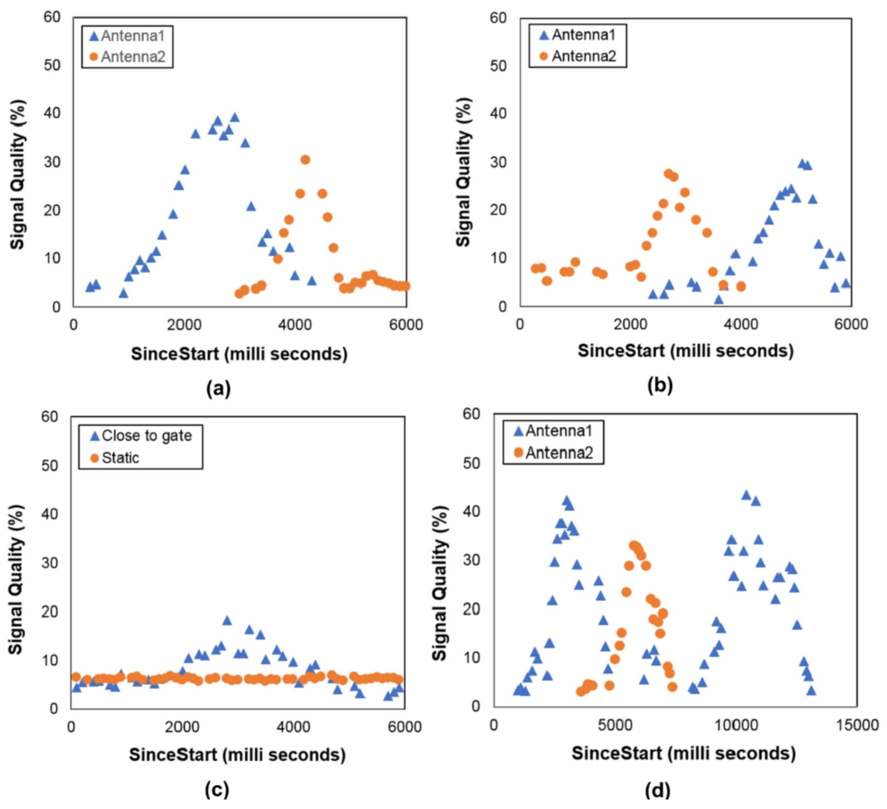


Figure 3: RSSI of the tags in the experimental setup of Alfian et al. [1].

within one zone have a smaller maximim RSSI. If tags move close to the gate and then away again, they are detected twice by the first antenna with a strong RSSI, and once by the second antenna with a slightly weaker RSSI.

Alfian et al. [1] used numerous machine learning algorithms. They found that some algorithms performed better than others, with the best performance being that of the XGBoost algorithm, and the poorest performance from the Multi-layer perceptron (MLP). Their results are shown in Table 1. The performance of the XGBoost

Table 1: Results obtained by Alfian et al. [1].

Algorithm	Accuracy	Precision	Recall	F1-score
MLP	63.59%	62.03%	59.13%	55.73%
Logistic regression	80.10%	80.08%	78.01%	77.45%
K-nearest neighbors	66.70%	64.35%	62.35%	62.32%
Decision tree	66.99%	61.89%	63.57%	58.58%
Naive Bayes	73.30%	75.22%	70.38%	70.51%
Random forest	92.72%	92.64%	92.04%	91.96%
AdaBoost	92.43%	92.13%	91.80%	91.64%
XGBoost	93.59%	93.25%	92.95%	92.78%

algorithm looks promising, but Mojix would like to see if they can get an even better performance.

In the problem statement it was mentioned that detecting false positives is also important since false positives have a negative impact on the accuracy of an RFID inventory management system. Ma et al. [7] investigated algorithms to identify false positives in an RFID readings. The data they used to do this, was more descriptive than the data used by Alfian et al. [1]. For example, they used summary statistics of individual tags' RSSI, such as, the maximum and minimum RSSI, the mean RSSI, the RSSI's range, variance and skewness, to give an idea. The purpose of this is because tags doing different movements, will give different RSSI patterns. For example, static tags will have a constant RSSI, meaning the maximum, minimum, and mean RSSI will be very close, and there will be very little variance in the RSSI. A tag that moves around in a zone will have a varying RSSI, but the maximum RSSI will be lower than the maximum RSSI of a tag that moves through the gate.

Ma et al. [7] compared three algorithms to detect false positives. The results are as follows:

- 92.75% accuracy for false positive detection using logistic regression.

- 95.30% accuracy for false positive detection using support vecor machine.
- 92.85% accuracy for false positive detection using decision tree.

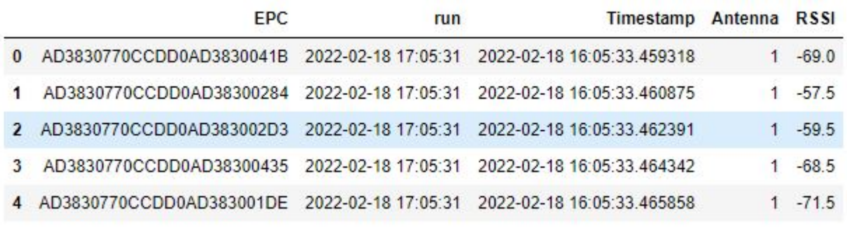
These results are promising, since no extra hardware was used to detect false positives. However, during the case study, Mojix came up with an idea that decreased false positives significantly. They proposed that the static tags be place at fixed distances from the dock door. They gathered data from runs with static tags at distances of 1m, 3m, and 5m away from the dock door. Using the distances of 3m and 5m effectively reduced the number of false positives without needing to use additional hardware. This is because by placing the tags further from the antennas, the RSSI of the static tags is significantly lower, and therefore, machine learning algorithms less frequently classify the static tags as moving tags.

3 Datasets

This section provides information on how the datasets used for the machine learning, look, but it does not go into fine details. It some of the final datasets were constructed in very complex ways, taking inputs from multiple other datasets.

3.1 Description

In order to analyze the movement of the tags through the dock door, a dataset containing a large amount of information has been put together. The data read by the antennas are structured in a table with 291 510 rows and 28 columns, with the most important ones for this case study being: 'Electronic Product Code (EPC)', 'run','Timestamp', 'Antenna', 'RSSI'. Figure 4 shows the first few rows of the initial dataset. Each RFID

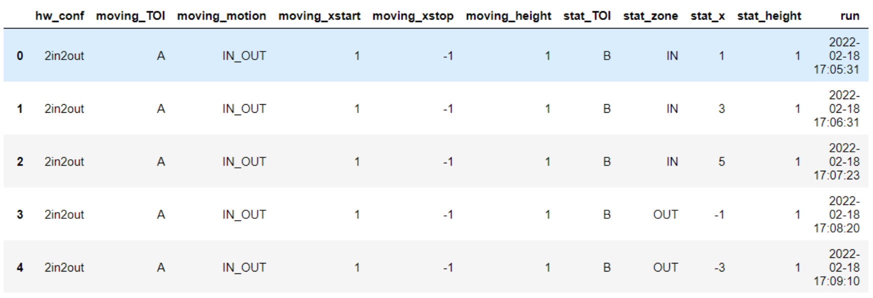


	EPC	run	Timestamp	Antenna	RSSI
0	AD3830770CCDD0AD3830041B	2022-02-18 17:05:31	2022-02-18 16:05:33.459318	1	-69.0
1	AD3830770CCDD0AD38300284	2022-02-18 17:05:31	2022-02-18 16:05:33.460875	1	-57.5
2	AD3830770CCDD0AD383002D3	2022-02-18 17:05:31	2022-02-18 16:05:33.462391	1	-59.5
3	AD3830770CCDD0AD38300435	2022-02-18 17:05:31	2022-02-18 16:05:33.464342	1	-68.5
4	AD3830770CCDD0AD383001DE	2022-02-18 17:05:31	2022-02-18 16:05:33.465858	1	-71.5

Figure 4: The first five rows of the dataset.

tag has a unique 'EPC', and this is the data that the antennas gather when a signal is received. The 'run' corresponds to when the recording of tag movements started. The time that a tag is detected by an antenna is recorded in the 'timestamp'. Then, the data is integrated from the servers attached to the antennas in a database. This data is recorded in milliseconds shown in the 'Timestamp' column. 'Antenna' corresponds to which antenna received the signal — 1 is the antenna located in the 'inside' zone and 2 is the antenna located in the 'outside' zone. 'RSSI' is the strength of the signal received by the antenna. Each antenna can, and does, read a unique EPCs multiple times. When the tag moves through the dock door, the other antenna will start to read the EPC more frequently.

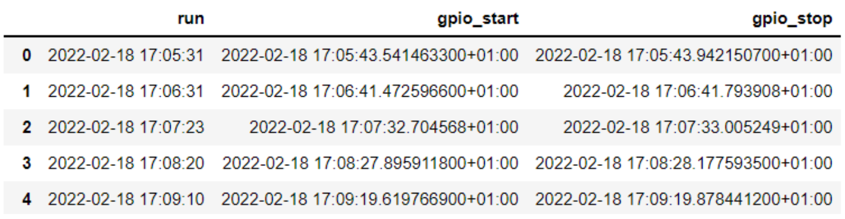
Next, there is a dataset containing information on each run. Since each run was concerned with one type of tag movement, and there are 72 runs, this dataset contains 72 rows. The most important columns of this dataset are the direction of the moving tags, from which zone to which zone the tags are moving, the distance inside the zones the motion starts and stops, in which zone the static tags are and also how far inside each zone the static tags are located. These columns are shown in Figure 5.



	hw_conf	moving_TOI	moving_motion	moving_xstart	moving_xstop	moving_height	stat_TOI	stat_zone	stat_x	stat_height	run
0	2in2out	A	IN_OUT	1	-1	1	B	IN	1	1	2022-02-18 17:05:31
1	2in2out	A	IN_OUT	1	-1	1	B	IN	3	1	2022-02-18 17:06:31
2	2in2out	A	IN_OUT	1	-1	1	B	IN	5	1	2022-02-18 17:07:23
3	2in2out	A	IN_OUT	1	-1	1	B	OUT	-1	1	2022-02-18 17:08:20
4	2in2out	A	IN_OUT	1	-1	1	B	OUT	-3	1	2022-02-18 17:09:10

Figure 5: The first five rows of the runs dataset.

Another necessary task was to determine when the tags moved out of one zone and into the other. Mojix came up with the idea to use a sensor that emits a constant signal at the dock door. The signal gets interrupted when the box or pallet with the tags crosses the line of signal, and the signal is received again when the box is done crossing the line of the signal. The sensor records the time of the signal interruption and resumption, and this then becomes the time that the tags moved from one zone to the other. These times were recorded in the gpio dataset, which is shown in Figure 6. The 'gpio_start' column indicates the timestamps the tags started



	run	gpio_start	gpio_stop
0	2022-02-18 17:05:31	2022-02-18 17:05:43.541463300+01:00	2022-02-18 17:05:43.942150700+01:00
1	2022-02-18 17:06:31	2022-02-18 17:06:41.472596600+01:00	2022-02-18 17:06:41.793908+01:00
2	2022-02-18 17:07:23	2022-02-18 17:07:32.704568+01:00	2022-02-18 17:07:33.005249+01:00
3	2022-02-18 17:08:20	2022-02-18 17:08:27.895911800+01:00	2022-02-18 17:08:28.177593500+01:00
4	2022-02-18 17:09:10	2022-02-18 17:09:19.619766900+01:00	2022-02-18 17:09:19.878441200+01:00

Figure 6: The first five rows of the gpio dataset.

crossing the dock door, and the 'gpio_stop' column indicates the timestamps the tags finished crossing the dock door.

All of these datasets were merged to create one dataset that contains all of the required information, *per tag*. Since there are multiple tags in one box or on one pallet, the movement for many tags are the same per run. The next subsection details some of the data preprocessing that was done.

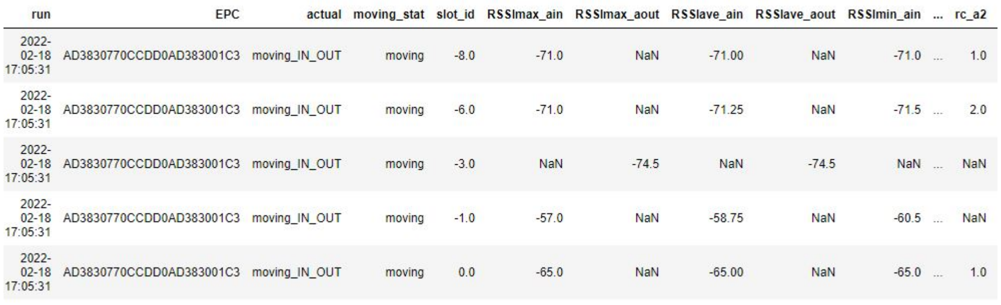
3.2 Cleaning and pre-processing

Cleaning and pre-processing the data is a process done before analyzing it. Indeed, within this step, new features are created to be used in the deep learning and machine learning algorithms detailed in the next section. RSSI depends on the distance between the antenna and the EPC tag. Higher RSSI signal indicate that the EPC tag is moving very close to the antenna. Therefore, the RSSI is a key element to identify the moving tags and the static ones. Table 2 shows a summary of the RSSI measurement data that were integrated into the previous data.

Table 2: Different RSSI attributes

Attribute Name	Description
RSSI MIN	The minimum of all RSSI values of a tag.
RSSI MAX	The maximum of all RSSI values of a tag.
RSSI AVE	The average of all RSSI values of a tag.
RSSI LINMAX	The maximum of all linear RSSI values of a tag.
RSSI LINAVE	The average of all linear RSSI values of a tag.
RSSI LINMIN	The minimum of all linear RSSI values of a tag.
Read count of the RSSI (RC)	The number of time an EPC is detected.

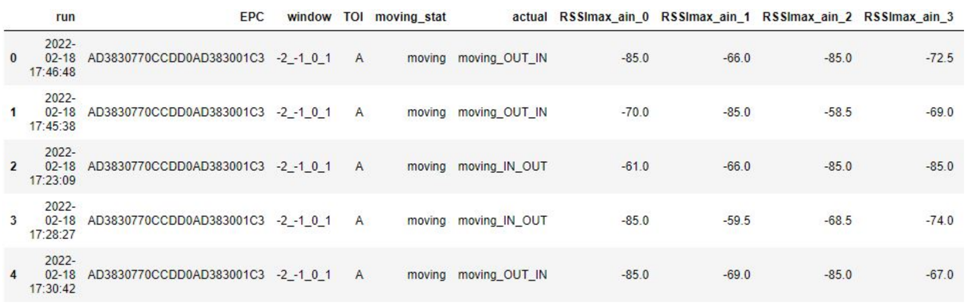
All of these 7 new features are concatenated and grouped by antenna coverage, to create in total 49 features that will be used by the machine learning models detailed in the next section *Algorithms*.



run	EPC	actual	moving_stat	slot_id	RSSImax_ain	RSSImax_aout	RSSIave_ain	RSSIave_aout	RSSImin_ain	...	rc_a2
2022-02-18 17:05:31	AD3830770CCDD0AD383001C3	moving_IN_OUT	moving	-8.0	-71.0	NaN	-71.00	NaN	-71.0	...	1.0
2022-02-18 17:05:31	AD3830770CCDD0AD383001C3	moving_IN_OUT	moving	-6.0	-71.0	NaN	-71.25	NaN	-71.5	...	2.0
2022-02-18 17:05:31	AD3830770CCDD0AD383001C3	moving_IN_OUT	moving	-3.0	NaN	-74.5	NaN	-74.5	NaN	...	NaN
2022-02-18 17:05:31	AD3830770CCDD0AD383001C3	moving_IN_OUT	moving	-1.0	-57.0	NaN	-58.75	NaN	-60.5	...	NaN
2022-02-18 17:05:31	AD3830770CCDD0AD383001C3	moving_IN_OUT	moving	0.0	-65.0	NaN	-65.00	NaN	-65.0	...	1.0

Figure 7: The first five rows of the dataset used in *Algorithms* section.

Next, a rolling window is joined to the table above. It shows one-second time slots per run. Each EPC might be detected in multiple time slots. For machine learning, we are more interested in keeping only the winning windows where the EPC is actually moving in/out through the dock door. Mojix has gathered the data in such a way that the crossing will always occur in the [-2,-1,0,1] time slot. Time slot 0 is where the crossing occurred, -2 is two seconds before the crossing, 1 is one second after the crossing etc. This means that the final data-frame contains the 49 features of each EPC during the [-2,-1,0,1] time slot, whether the EPC is moving or static. In addition, a binary classification of *moving* or *static* is applied to each EPC. This classification is the target feature of the predictive models explained in the *Algorithms* section.



	run	EPC	window	TOI	moving_stat	actual	RSSImax_ain_0	RSSImax_ain_1	RSSImax_ain_2	RSSImax_ain_3
0	2022-02-18 17:46:48	AD3830770CCDD0AD383001C3	-2_-1_0_1	A	moving	moving_OUT_IN	-85.0	-66.0	-85.0	-72.5
1	2022-02-18 17:45:38	AD3830770CCDD0AD383001C3	-2_-1_0_1	A	moving	moving_OUT_IN	-70.0	-85.0	-58.5	-69.0
2	2022-02-18 17:23:09	AD3830770CCDD0AD383001C3	-2_-1_0_1	A	moving	moving_IN_OUT	-61.0	-66.0	-85.0	-85.0
3	2022-02-18 17:28:27	AD3830770CCDD0AD383001C3	-2_-1_0_1	A	moving	moving_IN_OUT	-85.0	-59.5	-68.5	-74.0
4	2022-02-18 17:30:42	AD3830770CCDD0AD383001C3	-2_-1_0_1	A	moving	moving_OUT_IN	-85.0	-69.0	-85.0	-67.0

Figure 8: The first 5 rows of the dataset with the wining crossing-windows used in *Algorithms* section.

4 Algorithms

All of the algorithms used in this case study are briefly explained in this section. Finer details about hyper-parameters are given in the *Results* section.

4.1 k-Nearest Neighbors

k-Nearest Neighbours (kNN) is a supervised machine learning algorithm that can be used for both regression and classification. With a given new observation, kNN can classify the new unlabelled observation by analysing k observations closest to the new observation. In classification, the new observation is assigned the class that most of the k-nearest neighbors belong to, and in regression the new observation is assigned the mean of the k-nearest neighbors. Therefore, k is a hyper-parameter that is fixed during the training of the algorithm [3].

4.2 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning algorithm which can also be used for classification and regression. It uses the kernel trick to transform the data into a higher dimension and then based on these transformations it finds an optimal decision boundary that maximises the distance between the different classes (in the case of classification. New observations are assigned the same class of the observations on the same side of the decision boundary [10].

4.3 XGBoost

XGBoost has recently became one of the most popular machine learning algorithms. It often provides good accuracies when other machine learning algorithms perform badly. It is often the go-to algorithm in modern-day machine learning problems. In some cases, XGBoost can be more efficient than the random forest, which was the state-of-the-art machine learning algorithm for very long. It is from the family of boosting algorithms. It works by training a fixed number of weak classifiers — often decision trees with one or two levels — and after one classifier is trained on the data, the next one is trained but preference is given to correctly classify the points that were wrongly classified by the previous classifiers [9].

4.4 Neural Network

Neural networks get their names because they emulate how the human brain works. Many neurons are interconnected and transmit information to each other. Neural networks in machine learning is a network of neurons. It starts with an input layer, and ends with an output layer, and has at least one hidden layer in between. All of these layers contain one or more neurons. Each hidden layer processes the data coming from the previous layer to predict a specific target value, with the input layer taking the raw data, and the output layer outputting the prediction. Neural networks may perform well when dealing with sophisticated data which generally does not perform too well with tranditional machine learning algorithms [5].

5 Results

In the final dataset that was set up for the machine learning, there are 2 136 rows and 251 columns. The normal `train_test_split` function was used to split the data. The testing data were 33% of the entire dataset and the training data were the remaining 67%. The final data also only includes the linear RSSI values since Mojix determined through an analysis that the linear data produce better results than the non-linear data.

5.1 Moving versus static tags

In this specific task, the column 'moving_stat' was the target feature since it contains information on if a tag is moving or static. The dataset is unbalanced on this feature, with 1 760 moving observations and 376 static observations.

To determine if the tags are moving or static, classification machine learning algorithms were used. Specifically, the kNN, SVM, XGBoost algorithms, and a neural network were used and their performance compared. For the kNN algorithm, k was set equal to 3 and 5, and the performance was compared. For the SVM, the C-parameter was set to 10, the gamma-parameter was set to 0.01, and a radial basis kernel was chosen. These values were established through a grid search. For XGBoost, the number of weak classifiers (single-level decision trees) was set to 100, and the learning rate was set to 1. For the neural network, the feature-values were scaled using the `MinMaxScaler` and the labels were one-hot-encoded. The network has an input layer, then 3 hidden layers with 32 neurons in each layer. A dropout rate of 0.25 was used to prevent overfitting. The Adam optimiser was used with a learning rate of 0.0001.

The performance of the algorithms are shown in Table 3. The kNN algorithm with k = 5 performs better

Table 3: Performances to predict if tags are moving or static.

Algorithm	Accuracy	Precision	Recall	F1-score	False positives	False negatives
kNN (k = 3)	94.18%	94.40%	94.18%	94.27%	25	16
kNN (k = 5)	94.61%	94.84%	94.61%	94.69%	24	14
SVM	97.45%	97.49%	97.45%	97.46%	11	7
XGBoost	97.87%	97.86%	97.87%	97.85%	4	11
Neural network	97.45%	97.47%	97.45%	97.46%	10	8

than for k = 3. The SVM outperforms the kNN algorithms in all of the metrics. The neural network's performance is the same as the SVM and also better than the kNN. The best performance was obtained by the XGBoost algorithm. Particular attention should be paid to the number of false positives and false negatives. False positives are static tags that were predicted to be moving and false negatives are moving tags that were predicted to be static. Both errors are equally bad in an inventory management setup, since it the higher the number of these errors are, the less accurate the inventory management system will be. Therefore, the best algorithm for this task, is the XGBoost algorithm. The confusion matrices for the algorithms and the neural network's training and validation loss graphs are shown in Figure 9 in Appendix A. The it can be seen that the performance of the algorithms are similar, and that the XGBoost algorithm outperforms all of the other algorithms.

It should be noted that though these results are good, it is still a bit lower than the ideal. In the next sub section, this classification problem is refined a bit by being more specific with the classes.

5.2 Direction of moving tags and location of static tags

In this specific task, the column 'actual' was the target feature since it contains information on the tags' movement and locations. The dataset is unbalanced on this feature, with 880 moving in to out observations, 880 moving out to in observations, 209 static in observations, and 167 static out observations. For this task, the same algorithms were used, but the classification problem is now a multi-label classification problem. The possible labels are:

- moving_in_out
- moving_out_in
- static_in
- static_out

The only algorithm that changed was the neural network. For this task it has an input layer, 3 hidden layers with 64 neurons each, and a dropout rate of 0.4 to prevent overfitting.

The results for this task is shown in Table 4. Similar as the previous task, the kNN algorithms perform the

Table 4: Performances to predict the direction of moving tags and the location of static tags.

Algorithm	Accuracy	Precision	Recall	F1-score
kNN (k = 3)	86.38%	86.49%	86.38%	86.41%
kNN (k = 5)	87.66%	87.89%	87.66%	87.72%
SVM	92.06%	92.53%	92.06%	92.14%
XGBoost	91.63%	91.76%	91.63%	91.64%
Neural network	92.06%	92.48%	92.06%	92.12%

worst. The XGBoost algorithm performs better than the kNN. For this task, the SVM and neural network give the best performances and their performances are very similar. The results are again not as good as desired, but various hyperparameters were tried, and these are the best results. The confusion matrices for the algorithms and the neural network's training and validation loss graphs are shown in Figure 10 in Appendix B. The it can be seen that the performance of the algorithms are similar, and that the SVM and neural network algorithms perform similar and outperform the other algorithms.

5.3 Time tags move through the dock door

The dataset used for this task contains 644 rows along with an analytically-determined (by Mojix) label called 'actual', with possible values: 'crossing' and 'no crossing'. The aim is for Mojix to further develop this task to include timestamps of each RSSI reading, and using those timestamps to predict when the tags crossed the dock door, and along with task 2, to determine if the tags moved from inside the warehouse to outside, or *vice versa*. The dataset is severely unbalanced with 32 'crossing' labels and 612 'no crossing' labels. After some preliminary experimentation, it was found that all of the algorithms simply predicted 'no crossing' for all of the data because of how imbalanced the feature is. To combat this problem, Synthetic Minority Oversampling TEchnique (SMOTE) was used to oversample the 'crossing' class in only the *training* dataset, after using `train_test_split` with a test size of 40%. After using SMOTE, the training data were balanced, containing 367 observations of both labels.

The performance was still not fantastic, because the testing data was still unbalanced, but there was a slight improvement. The results are shown in Table 5. Even though the performance metrics are all close to or above

Table 5: Performances to predict if tags crossed the dock door.

Algorithm	Accuracy	Precision	Recall	F1-score	False positives	False negatives
kNN (k = 3)	90%	60%	69%	63%	19	7
kNN (k = 5)	89%	60%	72%	63%	23	6
SVM	95%	47%	50%	49%	0	13
XGBoost	96%	86%	61%	67%	1	10
Neural network	79%	53%	60%	52%	46	8

90%, it should be noted that there were only 13 positive (crossing) labels in the testing data, so by looking at the false positives and false negatives numbers, it is evident that the performance is not good. The confusion matrices of the algorithms and the loss graph of the neural network are shown in Figure 11 in Appendix C. It is evident that the SVM is not at all able to determine when tags cross the dock door, and the XGBoost and neural network cannot always determine when a crossing occurred. The kNN algorithms have more true positives, but at the expense of much more false positives. This task's definition likely still needs to be defined better to be able to obtain better results.

6 Conclusion and recommendations

In this last section we briefly discuss the conclusions that were made during the case study. We also give some recommendations that Mojix can consider for future work on this case study if they want to.

6.1 Conclusion

The first two tasks, determining if tags are moving or static and determining the direction of movement or location for static tags, are well defined, and similar porblems have been investigated in literature. Specifically for the first task, determining if tags are moving or static, the results obtained are similar to those obtained in the study done by Alfian et al. [1], which is that XGBoost has the best performance, but the SVM's and neural network's performances are very comparable to XGBoost's performance in this case study.

The second task, determining the direction of movement or location for static tags, is similar to the problem in the study by Alfian et al. [1], but the results obtained in this case study is different. In this case study the neural network and SVM had superior performances, whereas Alfian et al. [1] found that their neural network had the worst performance, and they also did not even evaluate an SVM. The first two tasks combined together should give Mojix fairly good results in predicting the movement and/or location of tags.

The third task needs some refining. The imbalance in the target feature makes all of the algorithms perform very poorly, even when the training data were oversampled. We argue that the results for this task is barely usable, but with some refining and a different dataset, we believe that very good results can be obtained.

All in all, it is concluded that machine learning is a good tool to use when working with RFID data because so much data is generated by the tags and readers. But as stated previously, problems need to be well-defined so that the appropriate data can be used to construct datasets for machine learning.

6.2 Recommendations

After working on this case study and reading some literature, we are able to make some recommendations to Mojix for future work on this topic. Firstly, as done by Alfian et al. [1], more types of movement can be captured when gathering training data. For example, rather than only gathering data for static tags and tags moving between zones, movement *within* zones can also be recorded. This will add additional noise in the data, but it is also more realistic than having only movement between two zones, and according to Alfian et al. [1], it is still possible to get good results when adding additional types of movements.

Secondly, the datasets are quite large in terms of the number of columns. It would be interesting to investigate which columns specifically influence the algorithms' performance. Columns that do not influence the performance can be seen as noise, and by removing the redundant columns, an improvement in performance will likely be observed. Due to our limited time and limited domain knowledge, we did not include this in our work.

Mojix did use a sensor that uses vision to determine when tags cross the dock door. Understandably, they do not want to keep this sensor there if it is not necessary, because it is something extra to pay for and monitor, it might break, and frequently needs to be cleaned to ensure accurate readings. But, if Mojix want better performance, combining the RFID data with this sensor's data might be a good solution.

Lastly, it was mentioned that the third task needs some refinement. We think that it could be a good task for regression if the data are gathered differently. It can be defined as a regression problem by taking the timestamp of the time of the crossing, and working out how many seconds elapsed from the start of the run. Regression can then be used to predict the time of the crossing. But, this would also need to be defined better, but it is an idea for Mojix for future work.

References

[1] Alfian, G., Syafrudin, M., Farooq, U., Ma'arif, M. R., Syaekhoni, M. A., Fitriyani, N. L., Lee, J., and Rhee, J. (2020). Improving efficiency of rfid-based traceability system for perishable food by utilizing iot sensors and machine learning model. *Food Control*, 110:107016.

[2] Atlas RFID Store (undated). What is RFID? — The Beginner's Guide to How RFID Systems Work. [Online] Available from: https://www.atlasrfidstore.com/rfid-beginners-guide/. [Accessed: 11 March 2022].

[3] Christopher, A. (2021). K-nearest neighbor. [Online] Available from: https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4#:~:text=K%2Dnearest%20neighbors%20(KNN),closet%20to%20the%20test%20data. [Accessed: 19 March 2022].

[4] Donaldson, J. (2022). Mojix: About Us. [Online] Available from: https://www.mojix.com/about-mojix/. [Accessed: 9 March 2022].

[5] Gudikandula, P. (2019). A beginner intro to neural networks. [Online] Available from: https://purnasaigudikandula.medium.com/a-beginner-intro-to-neural-networks-543267bda3c8.

[6] Hilyer, R. (2018). RSSI's Role in RFID. [Online] Available from: https://www.atlasrfidstore.com/rfid-insider/rssi-role-rfid. [Accessed: 11 March 2022].

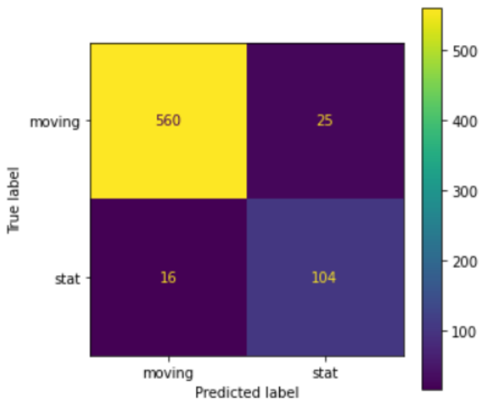
[7] Ma, H., Wang, Y., and Wang, K. (2018). Automatic detection of false positive rfid readings using machine learning algorithms. *Expert Systems with Applications*, 91:442–451.

[8] Niero, G. (2022). Mojix presentation (eng.). [Online] Available from: https://www.slideshare.net/gniero/mojix-presentation-eng.

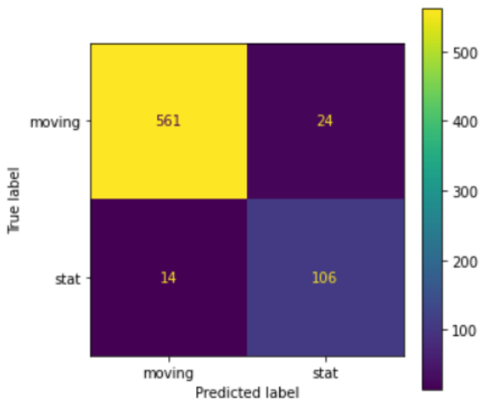
[9] Seif, G. (2022). A beginner's guide to xgboost. [Online] Available from: https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7.

[10] Yadav, A. (2018). Support vector machines(svm). [Online] Available from: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589. [Accessed: 11 March 2022].

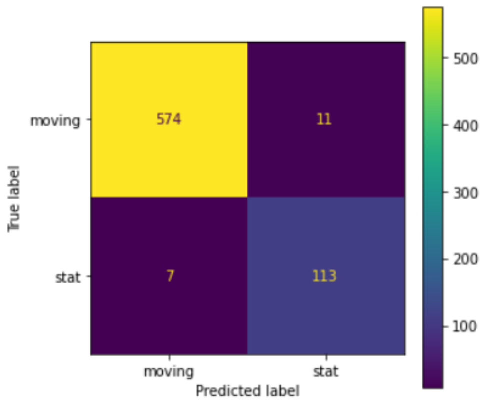
A The confusion matrices for Task 1



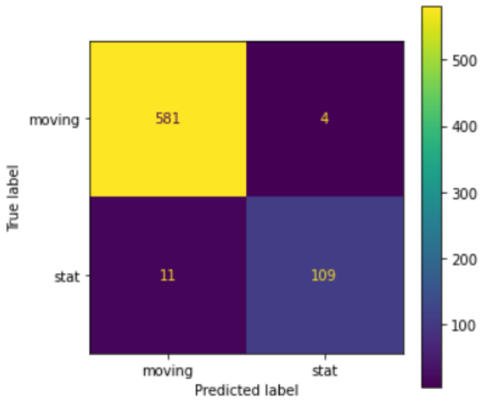
(a) Confusion matrix for the kNN (k = 3).



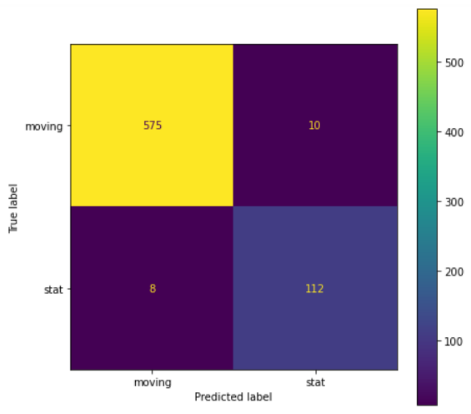
(b) Confusion matrix for the kNN (k = 5).



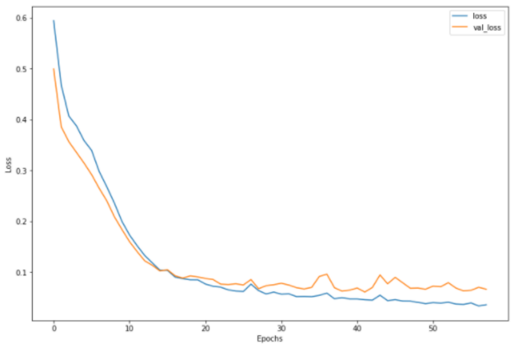
(c) Confusion matrix for the SVM.



(d) Confusion matrix for XGBoost.



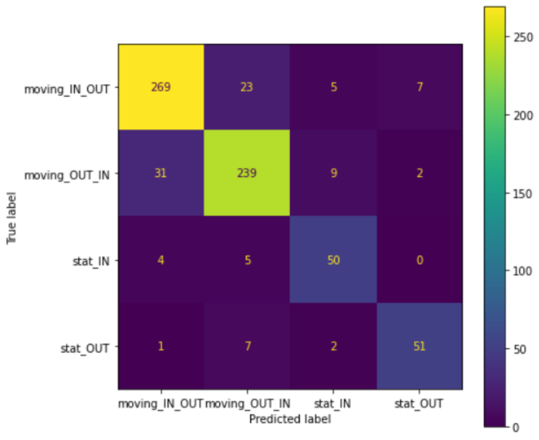
(e) Confusion matrix for the neural network.



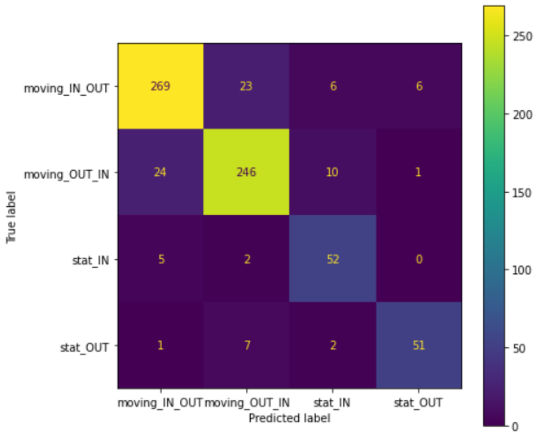
(f) Loss graph for the neural network.

Figure 9: Confusion matrices for the algorithms for task 1.

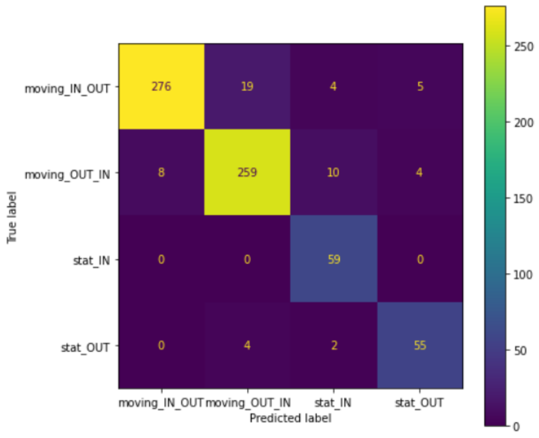
B The confusion matrices for Task 2



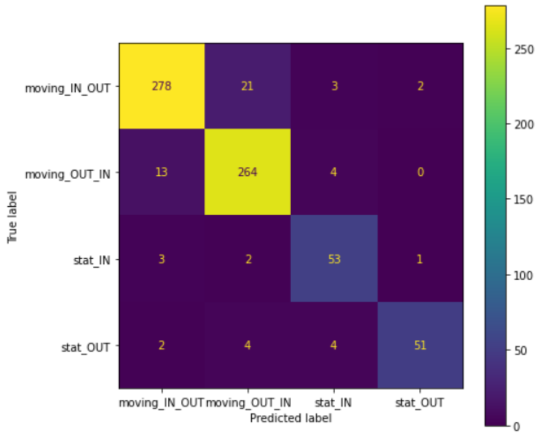
(a) Confusion matrix for the kNN (k = 3).



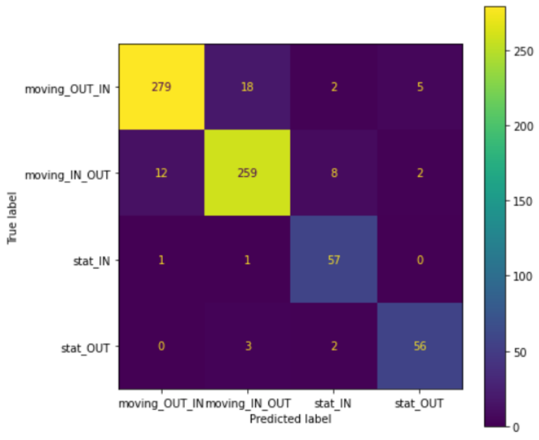
(b) Confusion matrix for the kNN (k = 5).



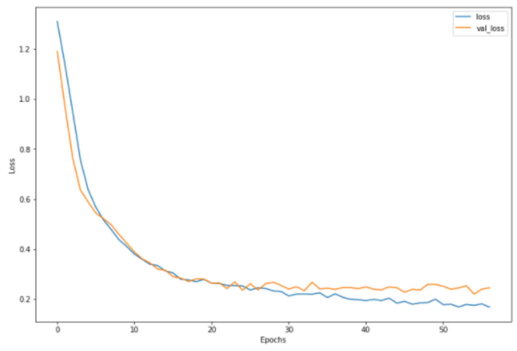
(c) Confusion matrix for the SVM.



(d) Confusion matrix for XGBoost.



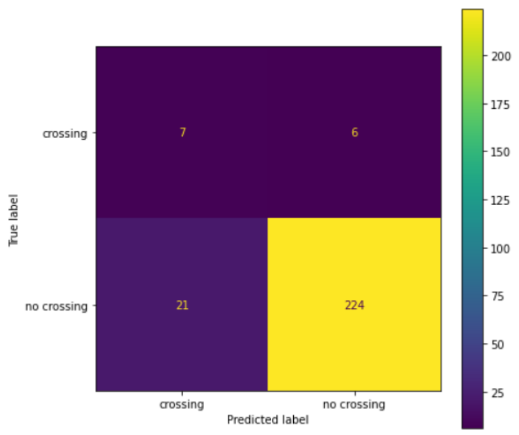
(e) Confusion matrix for the neural network.



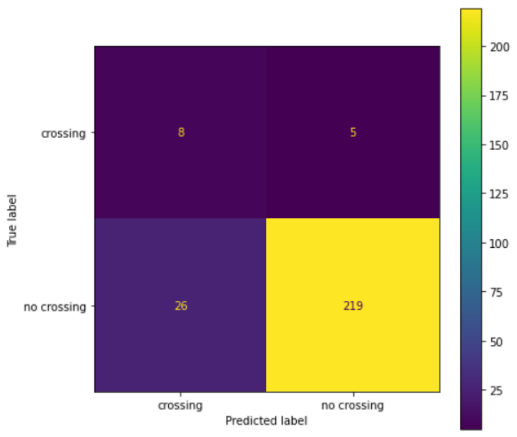
(f) Loss graph for the neural network.

Figure 10: Confusion matrices for the algorithms for task 2.

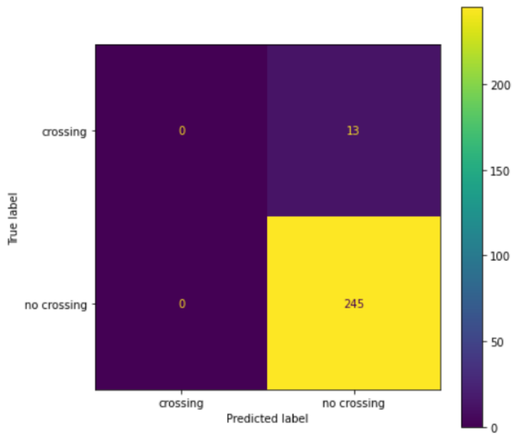
C The confusion matrices for Task 3



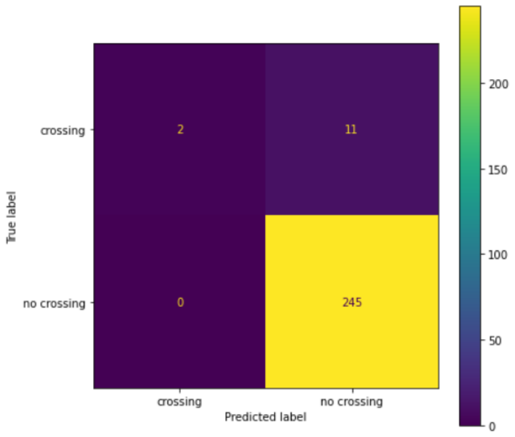
(a) Confusion matrix for the kNN (k = 3).



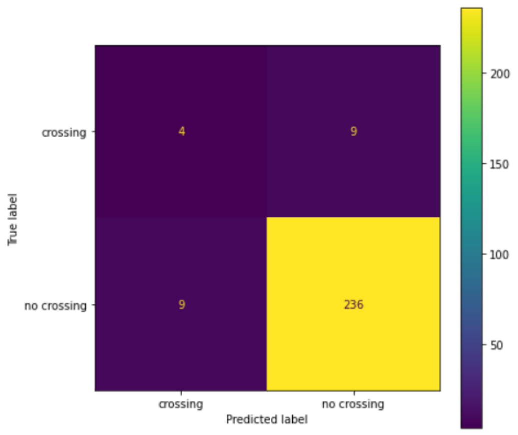
(b) Confusion matrix for the kNN (k = 5).



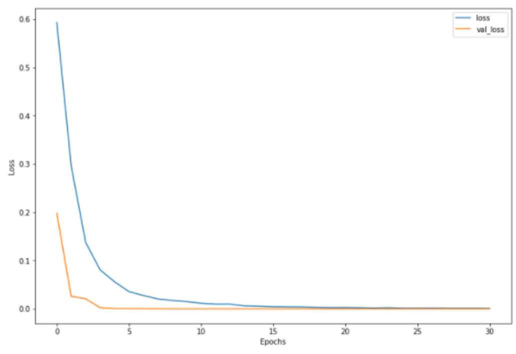
(c) Confusion matrix for the SVM.



(d) Confusion matrix for XGBoost.



(e) Confusion matrix for the neural network.



(f) Loss graph for the neural network.

Figure 11: Confusion matrices for the algorithms for task 3.