Expert Systems With Applications 184 (2021) 115562

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Real-time machine learning-based approach for pothole detection

Oche Alexander Egaji [a,1,*], Gareth Evans [a], Mark Graham Griffiths [a], Gregory Islas [b]

[a] Centre of Excellence in Mobile and Emerging Technologies (CEMET), Faculty of Computing, Engineering and Science, University of South Wales, Pontypridd, UK
[b] Mobilized Construction, Brunel House 1st Floor, 2 Fitzalan Rd, Cardiff, UK

ARTICLE INFO

Keywords:
Pothole detection
Machine learning
Vibration-based analysis
Accelerometer and Gyroscope
K-fold cross-validation

ABSTRACT

Potholes are symptoms of a poorly maintained road, pointing to an underlying structural issue. A vehicle's impact with a pothole not only makes for an uncomfortable journey, but it can also cause damage to the vehicle's wheels, tyres and suspension system resulting in high repair bills. This study presents a comparative study of machine learning models for pothole detection. The data was collected from multiple android devices/routes/cars and pre-processed using a 2-second non-overlapping moving window to extract relevant statistical features for training a binary classifier. The Test dataset was isolated entirely from the Training and Validation datasets, and a stratified K-fold cross-validation was applied to the Training dataset. The Random Forest Tree and KNN showed the best performance on the Test dataset with a similar accuracy of 0.8889. The model performance increased when random search hyperparameter tuning was applied to optimise the Random Forest Tree model's hyperparameters. The Random Forest Tree model's performance after hyperparameter tuning is 0.9444, 1.0000, 0.8889 and 0.9412 for accuracy, precision, recall, and F-score, respectively.

1. Introduction

Potholes are symptoms of a poorly maintained road, which could point to an underlying structural issue. A vehicle's impact with a pothole not only makes for an uncomfortable journey, but it can also cause damage to the vehicle's wheels, tyres and suspension system resulting in high repair bills. A recent survey showed an increase of 24% in the number of filled potholes in England and Wales (Asphalt Industry Alliance, 2019). Potholes account for a third of mechanical issues on the UK roads, and they cause the British motorist £2.8 billion every year (Asphalt Industry Alliance, 2013). The amount paid in road user compensation in England and Wales is £6.9 m, which excludes the £19.8 m staff cost for settling claims. About 89% of claims in England are because of potholes damages, which is up by 80% from 2018 (Asphalt Industry Alliance, 2019).

These figures are alarming and can be significantly reduced by investing in appropriate pothole detection technologies that make the reporting process seamless. In a recent RAC Business survey involving 500 UK companies, 46% of bosses reported the poor state of the roads and the cost associated with repairing a vehicle damaged by potholes is a significant challenge for business (RAC Business, 2016). Of more concern, the effect of potholes on pedestrians, cyclists and motorbike riders could be much more severe as it could lead to personal injuries. Research by Cycling UK suggests that 56% of people say they would cycle more if roads had fewer faults, such as potholes (Jones, 2018). Hence, the detection and reporting of potholes is key to ensuring the appropriate authorities are aware of the issue's scale. This will ensure that appropriate repairs are carried out in a timely manner.

The rapid technological advancement in recent years has led to miniaturisation and incorporation of robust sensors such as gyroscope, accelerometer, GPS, electronic compass, microphone, camera, etc., into mobile devices. This is the cheapest and most efficient way of pothole detection, as most people have a mobile phone, and there is no need for the installation of specialised hardware. This study utilised data collected from mobile devices in real-time to detect potholes. Most existing pothole detection approaches rely on specialised and expensive hardware devices, have lower accuracy or are not robust enough to detect real-world potholes.

This paper contains six sections. Section 2 includes the related work relevant to this study. Section 3 describes the machine learning models, whilst the research methodology is presented in Section 4. The results and analysis are discussed in Section 5. Finally, the conclusion and

* Corresponding author.
E-mail addresses: alexander.egaji@southwales.ac.uk (O.A. Egaji), gareth.evans@southwales.ac.uk (G. Evans), mark.griffiths@southwales.ac.uk (M.G. Griffiths), gregory@mobilizedconstruction.com (G. Islas).
[1] ORCID ID: 0000-0001-6661-7415.

https://doi.org/10.1016/j.eswa.2021.115562
Received 18 March 2020; Received in revised form 25 May 2021; Accepted 3 July 2021
Available online 10 July 2021

future work are shown in Section 6.

2. Related work

Several pothole detections approaches have been proposed over the last couple of years. According to Kulkarni et al. (2014), the existing pothole detection approaches can be classified into vibration-based methods, 3D reconstruction-based methods and vision-based methods.

The vibration-based approached involves the use of accelerometers, most often from a mobile device. This approach was used by Mednis et al. (2011); the authors compared the performance of multiple threshold-based approaches based on z-axis mobile sensing data. The authors claim a true positive rate as high as 90% for a small dataset. Also, Wang et al. (2015) and Madli et al. (2015) applied a threshold-based approach.

Fox et al. (2015) utilised a machine learning-based pothole detection approach. This approach relies on extracted features from crowdsourcing under-sampled simulated vehicles sensor data using CarSim. The authors claimed a simulated accuracy of 99.6% and an empirical experiment accuracy of 88.9% based on the simulated model. The authors' choice of aggregating simulated data from 500 vehicles will be challenging to replicate in the real world. There are additional GPS error issues, missing data, varying sensor configurations, and difficulty generalising the simulation platform, as shown in the accuracy degradation to 88.9% on a single stretch of road. A comparison of two machine learning models (SVM and gradient boosting) was carried out by Bhatt et al. (2017). The authors collected 21,300 observations of accelerometer and gyroscope with 96-labelled potholes from a single car using an iPhone 6Ss. They claim that SVM with an RBF kernel and gradient boosting achieved the best accuracy of 92.9% and 92.02%, respectively. However, the achieved precision (0.78) and recall (0.42) are much lower.

The image/vision-based approach involves using cameras (images or videos) for pothole data collection. Authors such as Zhang et al. (2014), Wang et al. (2017), Jo and Ryu (2015), Ouma and Hahn (2017), Ryu et al. (2015), Koch et al. (2013), Li Shuai et al. (2016) and Youquan et al. (2011) have used various image processing approaches on a small data sample for pothole detection. Other authors, such as Anand et al. (2018), have combined texture and spatial features of a camera image to train a deep neural network. The model was evaluated with 969 images. The resulting precision, recall and F-score are 92.4, 93.8 and 93.0%, respectively. However, this approach is computational complex rendering it less practical for real-time detection.

Most of the work in this area has used an image-based approach for pothole detection. They rely on libraries of actual potholes; hence, any variation in the pothole size, road markings, or even the presence of dirt on the road can affect the non-machine learning model's accuracy. This approach requires high computational power because of its computational complexity. Hence, it is not suitable for real-time pothole detection. The threshold-based model provides a simplified approach for the detection of a pothole. However, the heuristic process of determining the thresholds is tedious and prone to human error. Moreover, there is a high likelihood of encountering difficulty in generalising the model when it is deployed to other kinds of road surfaces or cars or the size of potholes. Most existing pothole detection approaches either rely on a specialised and expensive hardware device, have lower accuracy in pothole detection, or are not robust enough to detect all kinds of potholes. Also, the performance metric relied upon by some of the models is accuracy. This can often be confused with better model performance. Some of the existing work presented a higher accuracy with much lower precision and recall scores, highlighting the challenges they faced efficiently detecting potholes.

This study presents a comparative study of machine learning models for pothole detection. The data was collected from multiple android devices/routes/cars and pre-processed using 2-second interval aggregated chunks of data to extract relevant statistical features for training a binary classifier.

3. Machine learning models

Machine learning is a mathematical algorithm that enables a computer to learn from example data. The most common types of machine learning models are supervised and unsupervised learning. The algorithm learns from an example of labelled data in supervised learning and learns from unlabelled data in unsupervised learning. This paper compares the performance of five classification models (Naïve Bayes, Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbour (KNN) and Random Forest Tree).

3.1. Naïve Bayes

Naïve Bayes is a simplified probabilistic machine-learning algorithm that is commonly used in a classification task. The model can also be referred to as a simplified Bayesian probability model (Russell & Norvig, n.d.). The Naïve Bayes model assumes that all input features are independent of each other. Hence, changing the value of one should not directly influence the other feature present in the model. This assumption rarely holds in real-world applications, which is the reason it is called Naïve (Chai et al., 2004).

3.2. Logistic Regression

Logistic regression is a probability-based machine-learning approach that is mainly used for binary classification problems. The algorithm is similar to a linear regression model; however, it uses a more sophisticated cost function called the 'logistic or sigmoid function'. The sigmoid function maps a given input (z) to an output ($S(z)$) that varies between 0 and 1 (Peng et al., 2002).

3.3. K-Nearest Neighbours

K-Nearest Neighbours (KNN) is a basic but effective non-parametric supervised machine learning algorithm that can solve either classification or regression problems. The underlying intuition behind KNN requires classifying an unidentified test sample to a class 'c' represented by the majority of its k nearest neighbours in the Training dataset (Okfalisa et al., 2017). This approach is sometimes called the voting KNN rule and can often outperform more sophisticated methods. The k-nearest neighbours of an unidentified test sample are determined by calculating the Euclidean distance between the training and test points (Sun & Huang, 2010).

3.4. Support vector Machine (SVM)

A support vector machine is an algorithm that forms part of a supervised machine-learning approach. The algorithm is used either for regression or classification problems. The algorithm functions by finding the hyperplane in an N-dimension space (N – number of features) that distinctively separate/classify data points. The hyperplane is a decision boundary that helps separate the different classes of the data points, and its dimension depends on the number of features. For example, in a two-class classification problem, the algorithm estimates the hyperplane that maximises the margin (the distance between the data points from both classes) and minimises a quantity proportional to the misclassification error. This maximises the robustness of the model in reducing the classification error. The trade-off between maximising the margin and achieving a low training/test error is controlled by the positive regularisation parameter 'C'. Also, SVM can be used for cases where linear separation is not possible by varying the 'kernels', resulting in a hyperplane/feature mapping with non-linear boundary (Tzotsos & Argialas, 2008).

3.5. Random Forest Tree

Random Forest Tree (tree-based ensemble model) is a combination of multiple decision tree predictors, where an individual tree relies on values of an independently sampled random vector with a similar distribution. The basic principle of an ensemble approach is having a group of weak learners (individual tree/decision tree) to form a strong learner (forest). The models are fast and easy to implement; the Random Forest Tree model's bootstrapping nature reduces the model's tendency to over-fit when dealing with significant input variables/data. The trees are grown using the CART methodology developed by Georganos et al. (2019). The final training outcome is a set of decision rules; this includes rules that decide how to split the data at the node and decide when the branch is terminal and can no longer be split. These rules can either be continuous (for regression tree) or categorical (for classification tree) and capable of predicting an outcome variable.

4. Methodology:

This study focused on identifying pothole and non-pothole events. Hence, it was a binary classification problem. The initial stage of the project involved the data collection and labelling of the pothole or non-pothole events. This led to the development of two bespoke android applications (apps). The first records the accelerometer, gyroscope and GPS data, and the second app help in data labelling. The raw sensor data from both apps were pre-processed, merged, cleansed, and split (Training/Validation and Test) before extracting relevant features ready to train with the machine learning model. The research methodology is shown in Fig. 1.

The devices selected were relatively cheap phones that were capable of running Android One (v9.0). The three selected android phones used for data collection and labelling are the Nokia 3.1/5.1 and the Motorola G7. An adhesive sticking pad was used to mount the first phone to the centre of the dashboard face up, so the positive and negative z-axis is faced upwards and downwards, respectively. The data collection covered five different routes, road surfaces and four cars (Toyota Yaris Hybrid (sedan), Kia Niro (SUV), Skoda Octavia Estate (sedan) and Hyundai Getz (sedan)).

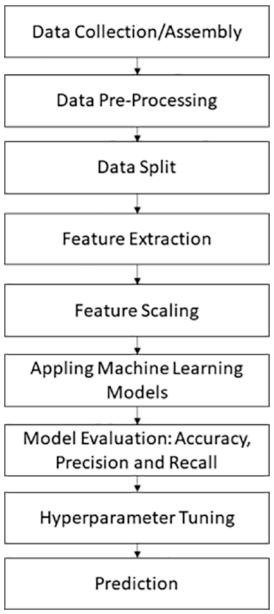

Fig. 1. Flow chart of research methodology.

4.1. Data Collection

The two developed bespoke apps ran on two different android phones. App 1, as shown in Fig. 2, records the UNIX timestamp, accelerometer data (x, y, z), gyroscope data (x, y, x) and location data (longitude and latitude). App 2, as shown in Fig. 3, records the GPS and UNIX timestamp data and is used for the manual recording of pothole events. Both devices have a sampling frequency of approximately 100 Hz. The device with app1 was mounted at the centre of the car's dashboard in a specific orientation; a passenger held the second device with app 2 in the car. The passenger was required to press and hold the blue button shown in Fig. 3 when the car approaches a pothole and release the button soon after the car has driven over the pothole. The app starts recording the sensor UNIX timestamp and GPS location when pressed until it is released. The authors took care to ensure that the clocks of both devices were synchronised. The recorded data was stored in the phone storage and uploaded to a database. The data from both phones were merged using the UNIX timestamp and the GPS location for extra validation.

4.2. Feature Extraction

The data collection process resulted in 10,913,863 observations of accelerometer, gyroscope, location data (longitude and latitude) and UNIX timestamp with 30,808-labelled potholes. The labelled potholes account for approximately 0.28% of the whole dataset, which is highly imbalanced. The data was collected at a high frequency; hence there is a likelihood of capturing measurements unrelated to the variables of interest and the measurement error resulting in noisy data points. Care was taken to minimise the noisy data points by grouping the data into 2-second interval chunks and calculating the aggregated statistical features for each interval chunk. This approach is similar to the non-overlapping moving window. The moving window computes the statistical features at time (t) by using a window that includes the raw data at a time (t) as well as past data up to time (t-2) seconds. The window is then shifted forward by time (t + 2), and the process continues until features for all the raw data are computed. The authors ensured there were no overlapping data chunks to avoid data leakage. Overall, an aggregate of 42 statistical features was extracted from each interval of the raw accelerometer and gyroscope data, as shown in Table 1.

4.3. Data Exploration

The time-domain of the z-axis for all the trips is shown in Fig. 4. This plot consists of the pothole and non-pothole events for all routes/cars used for data collection. The abrupt changes in values seen in the time domain can be due to noisy signals from the sensors. The signal noise can be noticed in the frequency domain shown in Fig. 5. A much cleaner signal can be seen in Fig. 6 after applying the 2-second aggregated statistical feature extraction on the raw z-axis data.

Fig. 2. App 1 – Data logger (Accelerometer, Gyroscope, GPS and UNIX timestamp).

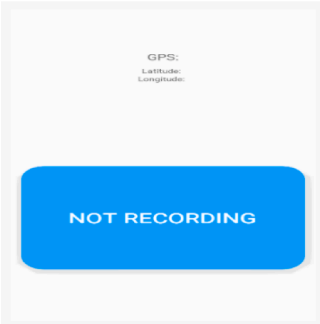

Fig. 3. App 2 – Data Logger (GPS and UNIX timestamp).

Table 1
Feature Extractions.

Name	Axis (Accelerometer and Gyroscope)	Domain
Minimum Value	X, Y, Z	Time
Maximum Value	X, Y, Z	Time
Mean	X, Y, Z	Time
Standard Deviation	X, Y, Z	Time
Variance	X, Y, Z	Time
Skewness	X, Y, Z	Time
Kurtosis	X, Y, Z	Time

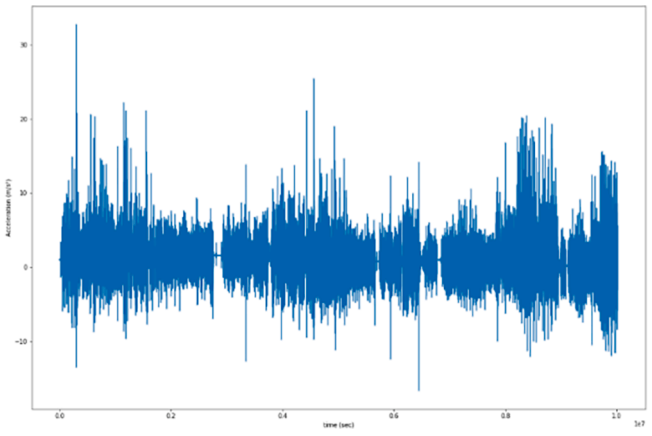

Fig. 4. Time Domain – Raw Z-axis data.

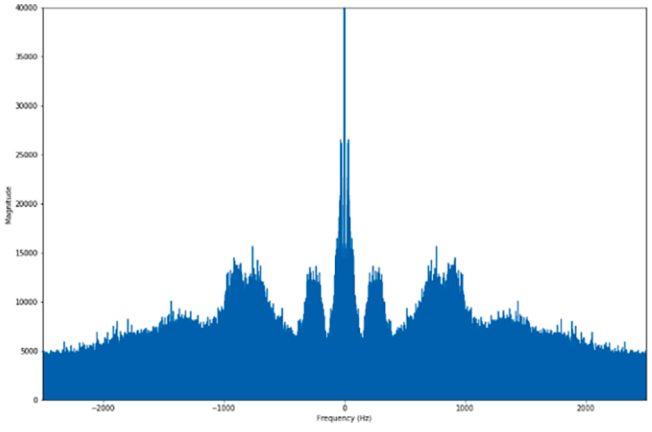

Fig. 5. Frequency Domain - Z-axis data.

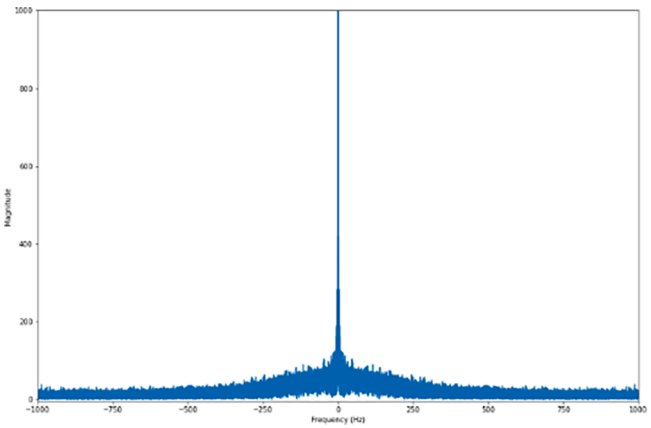

Fig. 6. Frequency Domain – Z-axis Mean (non-overlapping 2-second Window).

4.4. Evaluation Metric and Cross-Validation

The performance metrics used for this paper are accuracy, precision, recall and F-score. The accuracy consists of the ratio of correctly predicted observations to total observations, and it is the commonly used predictive model evaluation metric. It is common to confuse high accuracy with better model performance; however, this can only be when the number of false-positive and false-negative events are almost the same for a balanced dataset. The precision is the ratio of the correctly predicted positive observation and the total predicted positive observation. The precision is a good measure when the cost of false positive is high. The recall is a measure of how much of the total actual positively labelled datasets the model captures. Recall is a good measure when there is a high cost associated with false-negative. A combination of the precision and recall metrics into a single metric that capture both properties can be represented by the F-score (Divya et al., 2019). The F-score gives an overall view of model performance so that neither precision nor recall can be considered individually.

The dataset was balanced by randomly and uniformly undersampling the majority class. This ensures that the overall data to be used for the machine-learning models have equal numbers of potholes and non-potholes samples. A vital step in developing a machine learning model and ensuring it generalises better is the cross-validation step. This will help to prevent developing a model with high bias or variance. The datasets collected were concatenated into a single data source to form a combined training/validation dataset, except for data from two routes/two cars which was separated to be used as a test dataset for the machine learning model. This was separated to avoid similarity bias and data leakage where information from the Test dataset is present in the Training dataset. As described in Section 4.2, the features extraction was applied separately to the combined (Training/Validation) and Test datasets.

A combination of holdout and the stratified K-fold cross-validation approach was used in this paper. The combined dataset was partition into training and validation datasets. The models were trained on the training dataset, and the Validation dataset was used to manually tune the machine learning models' hyperparameters. This approach utilises a different combination of the hyperparameter set based on experience to tune the model. The performance was relatively similar to the model with the default parameters (scikit-learn python libraries (Pedregosa et al., 2011)). Afterwards, the Test dataset is used to evaluate how well the models perform on unseen data. The current data has been split into 80% Train, 10% Validation and 10% Test. A stratified K-fold cross-validation was applied to the Training dataset portion of the split with $K = 10$. The K-fold cross-validation randomly partitioned the data into K smaller sets. One of the K-subsets is used to evaluate the model, and the other (K-1) subsets are used to train the model. The process is repeated K

times until all unique subsets have been used to evaluate the model. The average accuracy and standard deviation can be computed across all K-sessions. This will give a better indication of the model performance and whether it overfits the data.

5. Result and analysis

The average accuracy at the 95% confidence interval ($\mu \pm 2\sigma$, where μ is the mean and σ the standard deviation) for the five machine learning models – Naïve Bayes, Logistic Regression, SVM, K-Nearest neighbour and Random Forest Tree with K = 10 is shown in Table 2. The SVM, Random Forest Tree and KNN are the best performing model with an average accuracy at a 95% confidence interval of 0.8200 ± 0.1598, 0.8071 ± 0.1259 and 0.7879 ± 0.1371, respectively. The KSVM, Random Forest Tree and KNN default Hyperparameters and their description for the trained model are given in Table 3. A detailed explanation of the hyperparameters can be found in the scikit-learn python libraries documentation (Pedregosa et al., 2011).

The Validation dataset was used to evaluate the manual parameter turning of the models. The performance of the machine-learning models on the Validation dataset is shown in Table 4. The Random Forest Tree and KNN are the two best performing models with accuracy, precision, recall and F-score of over 77%. According to the precision, recall, F-score and AUC metrics in Table 4, the Random Forest Tree and KNN are the best performing models amongst the five considered models. The accuracy, precision, recall, F-score and AUC for the Random Forest Tree and KNN are 0.8889, 0.8571, 0.8571, 0.8831 and 0.8889, and 0.7778, 1, 0.8750 and 0.9091 respectively. The recall of the KNN is perfect at 1.00 – meaning the model has found all the actual potholes that can be found in the Validation dataset. However, the lower precision is because of the many false positives (non-pothole data points classified as a pothole by the model).

The Test dataset was used to evaluate the model's performance on totally unseen data. The performance of the machine-learning models on the Test dataset is shown in Table 5. Except for the Logistic Regression, all the models have a Test dataset accuracy that is over 83%. The performance metrics for the Naïve Bayes and KNN are the same for the accuracy and AUC and both have similar F-scores. The Random Forest Tree model performed the best with Test dataset accuracy, precision, recall, F-score and AUC of 0.8889, 1.0000, 0.7778, 0.8750 and 0.8889, respectively. The Random Forest Tree precision is a perfect score of 1.0000 – this implies that when the model predicts a pothole, it is correct 100% of the time. However, a lower recall score of 0.8889 is because of the many false negatives (i.e., potholes data points classified as non-potholes).

5.1. Hyperparameter tuning of Random Forest Tree using Random Search

The Random Forest Tree model is one of the best performing models in the previous section using the hyperparameters shown in Table 3. The selected hyperparameters are the best starting point for tuning the model's hyperparameter for better performance. This section explores the impact of using the random search hyperparameter tuning on the Random Forest Tree model's performance. This hyperparameter tuning approach is beneficial because it does not use the entire hyperparameter combination. Instead, it used a random combination of the initialised grid of hyperparameter values to train and score a model. The number of search iteration can be set based on the available resources and time (Bergstra & Bengio, 2012). The initialised grid of hyperparameter values is shown in Table 6, and the search was run for 100 iterations. The same Training and Validation datasets used in the previous section were used

Table 2
Performance Metric – K-Fold Cross-Validation (K = 10) Training Dataset.

Models	Average Accuracy (95% Confidence interval)
Naive Bayes	0.7758 ± 0.0985
Logistic Regression	0.7825 ± 0.1345
SVM	0.8200 ± 0.1598
KNN	0.7879 ± 0.1371
Random Forest Tree	0.8071 ± 0.1259

Table 3
Random Forest Tree and K-Nearest Neighbours Hyperparameter Parameter.

Machine Learning Model	Parameters and Values	Parameters Description
SVM	Kernal = rbf	Specifies the kernel type to be used in the algorithm
	Probability = True	Enables probability estimates
	Random_state = 0	Controls the pseudo-random number generation
Random Forest Tree	n_estimators: 100	Sets the number of decision trees to be used in the forest
	min_samples_split: 2	Minimum number of samples needed before a split
	min_samples_leaf: 1	Minimum number of samples needed to create a leaf
	max_features: auto	Number of features to consider for the best node split
	max_depth: None	Maximum depth of the tree
	Criterion: gini	Measures the quality of a split
	Bootstrap: True	Specify if bootstrap samples are used when building trees
	Random_state = 0	Controls the randomness of the bootstrapping of the samples
K-Nearest Neighbours	n_neighbor: 5	Number of neighbours to use for neighbours' queries
	weights: uniform	The weight function used in the prediction
	Algorithm: auto	Use to compute the nearest neighbours
	Metric: Minkowski	The distance metric to use for the tree
	P: 2	Power parameter for the Minkowski metric

Table 4
Performance Metric – Validation Dataset.

Models	Accuracy	Precision	Recall	F-score	AUC
Naive Bayes	0.8333	0.7500	0.8571	0.8000	0.8377
Logistic Regression	0.8333	0.7500	0.8571	0.8000	0.8377
KSVM	0.8333	0.7500	0.8571	0.8000	0.8377
KNN	0.8889	0.7778	1.0000	0.8750	0.9091
Random Forest Tree	0.8889	0.8571	0.8571	0.8571	0.8831

Table 5
Performance Metric – Test Dataset.

Models	Accuracy	Precision	Recall	F-score	AUC
Naive Bayes	0.8889	1.0000	0.7778	0.8750	0.8889
Logistic Regression	0.6667	0.7143	0.5556	0.6250	0.6667
KSVM	0.8333	1.0000	0.6667	0.8000	0.8333
KNN	0.8889	0.8889	0.8889	0.8889	0.8889
Random Forest Tree	0.8889	1.0000	0.7778	0.8750	0.8889

Table 6
Initialised Grid of Hyperparameter for Random Forest Tree Model.

Parameters	Values
n_estimators	100–2000
min_samples_split	2, 5, 10, 15
min_samples_leaf	1, 2, 4, 8
max_features	auto, sqrt
max_depth	10–120
Criterion	gini, entropy
bootstrap	True, False

for the random search model. The optimum hyperparameters for the Random Forest Tree model after tuning with random search is shown in Table 7. There was a 6.2500% improvement in the Validation dataset's accuracy after the hyperparameters tuning, as shown in Table 8. The percentage improvement after hyperparameter tuning on the Validation dataset for accuracy, precision, recall, F-score and AUC is 6.2487, 2.0884, 16.6725, 8.8943 and 8.0903%, respectively. The optimised model was evaluated further on the Test dataset, and the results are shown in Table 9. The percentage improvement after hyperparameter tuning on the Test dataset for accuracy, precision, recall, F-score and AUC are 6.2500, 0.0000, 14.2857, 7.5630 and 6.2500%, respectively.

The precision-recall curve for the optimised Random Forest Tree model on the Test data is shown in Fig. 7. The curve shows the relationship between precision and recalls at every possible threshold/cut-off. The cut-off values determine the fraction of true positive or true negative prediction of the model. The perfect performance is a model that can discriminate between pothole and non-pothole with 100% recall and 100% precision. Hence, the graph line will pass through the top left corner (0.0, 1.0) and top right corner (1.0, 1.0). The curve that is closer to the perfect precision curve has a better performance than the baseline. The Random Forest Tree model's chosen cut-off led to accuracy, precision, recall and F-score of 0.9444, 1.0000, 0.8889 and 0.9412, respectively.

5.2. Overview

The model was evaluated on data completely isolated from the Training and Validation datasets. The Validation datasets were used to evaluate the performance of the hyperparameter tuning. The Random Forest Tree and KNN models showed the best performance on the Test dataset compared to the other models before tuning the hyperparameter. The random search hyperparameter tuning approach was then applied to the Random Forest Tree model, and it furthered increased the model's performance. In addition, the model generalises better to unseen data, which contradicts the behaviour noticed by Fox et al. (2015). This might have resulted from the use of an overlapping sliding window for feature extraction, which can cause Train-Test contamination as the pattern used for testing is not distinguishable from the training. This phenomenon was referred to as similarity bias by Rauber et al. (2021). It can overly simplify the model for generalising real-world problems.

6. Conclusion

This paper developed an intelligent pothole detection system using data collected from mobile sensors. The data was pre-processed to extract statistical features using a 2-second non-overlapping moving window.

A comparative study using five binary classification machine learning models (Naïve Bayes, Logistic regression, SVM, KNN and Random Forest Tree) was performed for balanced data. The Training and Validation datasets were isolated entirely from the Test dataset before feature extractions. A 2-second non-overlapping moving window was used for feature extraction to avoid similarity bias in the Training/Validation and Test data. A stratified K-fold cross-validation technique was applied to the training dataset with K = 10.

Table 7
Optimum Hyperparameter for Random Forest Tree parameter.

Parameters	Values
n_estimators	311
min_samples_split	5
min_samples_leaf	1
max_features	sqrt
max_depth	46
Criterion	entropy
bootstrap	False

Table 8
Performance Metric – Validation Dataset.

Models	Accuracy	Precision	Recall	F-score	AUC
Random Forest Tree	0.9444	0.8750	1.0000	0.9333	0.9545
Percentage Improvement (%)	6.2487	2.0884	16.6725	8.8943	8.0903

Table 9
Performance Metric– Test Dataset.

Models	Accuracy	Precision	Recall	F-score	AUC
Random Forest Tree	0.9444	1.0000	0.8889	0.9412	0.9444
Percentage Improvement (%)	6.2500	0.0000	14.2857	7.5630	6.2500

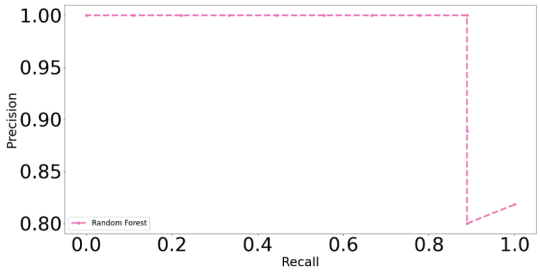

Fig. 7. Precision-Recall Curve (Test Data).

The Random Forest Tree and KNN showed the best performance on the Test dataset with a similar accuracy of 0.8889. This performance increased when random search hyperparameter tuning was used to optimised the Random Forest Tree model. The Random Forest Tree model's performance after hyperparameter tuning is 0.9444, 1.0000, 0.8889 and 0.9412 for accuracy, precision, recall, and F-score. The percentage improvement after hyperparameter tuning on the Test dataset for accuracy, precision, recall, F-score and AUC are 6.2500, 0.0000, 14.2857, 7.5630 and 6.2500, respectively. The Random Forest Tree model's precision is a perfect score of 1.0000 – this implies that when the model predicts a pothole, it is correct 100% of the time. However, a lower recall score of 0.8889 is because of the many false negatives (i.e., potholes data points classified as non-potholes). Hence, the model performed well on a new route/car, which was not part of the Training data.

However, sufficient samples from all roads and cars are required to evaluate further and build models that perform well on diverse road/car types. In addition, further annotation will be needed to help build a model that can categorise the potholes in more detail. This will enable road maintenance agencies to prioritise pothole fixing based on their severity.

CRediT authorship contribution statement

Oche Alexander Egaji: . **:** Conceptualization, Data curation, Methodology, Formal analysis, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Gareth Evans:** Conceptualisation, Data Curation , Visualisation, Software, Validation. **Mark Graham Griffiths:** Supervision, Funding acquisition. **Gregory Islas:** Conceptualization, Data curation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors will like to acknowledge the European Regional Development Fund (ERDF) and the Welsh Government for funding this study (WEFO 82127 &WEFO 80849). We will also like to acknowledge the contribution of Mobilized Construction in the data collection process phase of the project. Finally, our gratitude goes to all members of the Centre of Excellence in Mobile and Emerging Technologies (CEMET), the University of South Wales, for their contribution in various capacity in this study.

References

Anand, S., Gupta, S., Darbari, V., & Kohli, S. (2018). Crack-pot: Autonomous road crack and pothole detection. *Digital Image Computing: Techniques and Applications (DICTA), 2018*, 1–6. https://doi.org/10.1109/DICTA.2018.8615819

Asphalt Industry Alliance. (2013). Annual Local Authority Road Maintenance (ALARM) Survey. HMPR Limited. https://www.asphaltuk.org/wp-content/uploads/ALARM_survey_2013.pdf.

Asphalt Industry Alliance. (2019). Annual Local Authority Road Maintenance (ALARM) Survey. Asphalt Industry Alliance. https://www.asphaltuk.org/wp-content/uploads/alarm-survey-2019-digital.pdf.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*(2).

Bhatt, U., Mani, S., Xi, E., & Kolter, J. Z. (2017). Intelligent pothole detection and road condition assessment. ArXiv Preprint ArXiv:1710.02595.

Chai, X., Deng, L., Yang, Q., & Ling, C. X. (2004). Test-cost sensitive naive bayes classification. In *Proceedings of the Fourth IEEE International Conference on Data Mining* (pp. 51–58).

Divya, K., & Pabitha, P. (2019). Analysing the competency of various decision trees towards community formation in multiple social networks. *International Conference on Communication and Signal Processing (ICCSP), 2019*, 0099–0103. https://doi.org/10.1109/ICCSP.2019.8698110

Fox, A., Kumar, B. V. K. V., Chen, J., & Bai, F. (2015). Crowdsourcing undersampled vehicular sensor data for pothole detection. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)* (pp. 515–523). https://doi.org/10.1109/SAHCN.2015.7338353

Georganos, S., Grippa, T., Gadiaga, A., Vanhuysse, S., Kalogirou, S., Lennert, M., & Linard, C. (2019). An application of geographical random forests for population estimation in dakar, senegal using very-high-resolution satellite imagery. *Joint Urban Remote Sensing Event (JURSE), 2019*, 1–4. https://doi.org/10.1109/JURSE.2019.8809049

Jo, Y., & Ryu, S. (2015). Pothole detection system using a black-box camera. *Sensors (Basel, Switzerland), 15*(11), 29316–29331. https://doi.org/10.3390/s151129316

Jones, S. (2018, May 31). Britain's potential cyclists put off cycling due to traffic conditions and potholes [Cycling UK]. Cycling UK. https://www.cyclinguk.org/press-release/britains-potential-cyclists-put-cycling-due-traffic-conditions-and-potholes.

Koch, C., Jog, G. M., & Brilakis, I. (2013). Automated pothole distress assessment using asphalt pavement video data. *Journal of Computing in Civil Engineering, 27*(4), 370–378.

Kulkarni, A., Mhalgi, N., Gurnani, S., & Giri, N. (2014). Pothole Detection System using Machine Learning on Android.

Madli, R., Hebbar, S., Pattar, P., & Golla, V. (2015). Automatic detection and notification of potholes and humps on roads to aid drivers. *IEEE Sensors Journal, 15*(8), 4313–4318. https://doi.org/10.1109/JSEN.2015.2417579

Mednis, A., Strazdins, G., Zviedris, R., Kanonirs, G., & Selavo, L. (2011). Real time pothole detection using android smartphones with accelerometers. 1–6.

Okfalisa, Gazalba, I., Mustakim, M., & Reza, N. G. I. (2017). Comparative analysis of k-nearest neighbor and modified k-nearest neighbor algorithm for data classification. 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), 294–298. doi: 10.1109/ICITISEE.2017.8285514.

Ouma, Y. O., & Hahn, M. (2017). Pothole detection on asphalt pavements from 2D-colour pothole images using fuzzy c-means clustering and morphological reconstruction. doi: 10.1016/j.autcon.2017.08.017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12(85), 2825–2830.

Peng, C.-Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An introduction to logistic regression analysis and reporting. *The Journal of Educational Research, 96*(1), 3–14. https://doi.org/10.1080/00220670209598786

RAC Business. (2016, June 16). Potholes and congestion top Brit bosses' business bugbears [Royal Automobile Club]. RAC. https://media.rac.co.uk/pressreleases/potholes-and-congestion-top-brit-bosses-business-bugbears-1442185.

Rauber, T. W., da Silva Loca, A. L., de Boldt, F. de A., Rodrigues, A. L., & Varejão, F. M. (2021). An experimental methodology to evaluate machine learning methods for fault diagnosis based on vibration signals. *Expert Systems with Applications, 167*, 114022. https://doi.org/10.1016/j.eswa.2020.114022

Russell, S., & Norvig, P. (n.d.). Artificial Intelligence A Modern Approach 3 edition. Pearson Education Limited.

Ryu, S.-K., Kim, T., & Kim, Y.-R. (2015). Image-based pothole detection system for ITS service and road management system [Research article]. *Mathematical Problems in Engineering, 2015*, 1–10. https://doi.org/10.1155/2015/968361

Shuai, L., Chenxi, Y., Donghai, L., & Hubo, C. (2016). Integrated processing of image and GPR data for automated pothole detection. *Journal of Computing in Civil Engineering, 30*(6), 04016015. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000582

Sun, S., & Huang, R. (2010). An adaptive k-nearest neighbor algorithm. 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, 1, 91–94. doi: 10.1109/FSKD.2010.5569740.

Tzotsos, A., & Argialas, D. (2008). Support Vector Machine Classification for Object-Based Image Analysis. In T. Blaschke, S. Lang, & G. J. Hay (Eds.), Object-based image analysis: Spatial concepts for knowledge-driven remote sensing applications (pp. 663–677). Springer. doi: 10.1007/978-3-540-77058-9_36.

Wang, H.-W., Chen, C.-H., Cheng, D.-Y., Lin, C.-H., & Lo, C.-C. (2015). A real-time pothole detection approach for intelligent transportation system. Mathematical Problems in Engineering, 2015.

Wang, P., Hu, Y., Dai, Y., & Tian, M. (2017). Asphalt pavement pothole detection and segmentation based on wavelet energy field. Mathematical Problems in Engineering, 2017.

Youquan, H., Jian, W., Hanxing, Q., Zhang, W., & Jianfang, X. (2011). A research of pavement potholes detection based on three-dimensional projection transformation. 2011 4th International Congress on Image and Signal Processing, 4, 1805–1808. doi: 10.1109/CISP.2011.6100646.

Zhang, Z., Ai, X., Chan, C. K., & Dahnoun, N. (2014). An efficient algorithm for pothole detection using stereo vision. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 564–568). https://doi.org/10.1109/ICASSP.2014.6853659