

Contents

- Abstract2
- Introduction2
 - 1. Problem Understanding2
 - 2. Predictive Modeling:2
 - 3. Route Optimization.....3
 - 4. Reinforcement Learning.....3
- Task 1: Understanding the Problem3
 - Importance of Delivery Optimization.....3
 - Role of AI in Solving the Problem.....3
 - Problem Classification.....3
 - Regression vs. Classification4
 - Conclusion4
- Task 2: Predictive Modeling.....4
 - Data Preprocessing4
 - Models Training4
 - Comparison7
- Task 3: Grid Cost Estimation and Pathfinding8
 - Data Processing8
 - Prediction using the LSTM9
 - Graphical Representation9
 - Comparison of Dijkstra’s and A*..... 10
- Task 4: Q-Learning for Smart City Navigation 12
 - Grid Environment and Q-Learning Setup 12
 - Hyperparameter Tuning and Convergence..... 13
 - Optimal Policy Evaluation 13
- Conclusion..... 15
- References..... 15

Abstract

This report seeks to answer the problem of delivering goods and services in an ever-evolving smart city setting through incorporating Artificial Intelligence (AI) strategies to produce satisfactory results that would reduce cost of delivery as well as polluting to the environment. The issue is solved using machine learning techniques, such as data preprocessing and algorithmization, as well as further city grid data analysis and implementation of the best route search algorithm. . Further, a method of applying the Q-learning algorithm is demonstrated for training an autonomous vehicle agent for grid-world navigation – an example of RL in practice. The results stress on the capacity of AI to solve intricate logistical challenges and propose the enhanced urban delivery network..

Introduction

On-time delivery and delivery period are among the critical factors in today's urban logistics helping businesses and city dwellers to obtain their stock in the shortest possible time. The specifics that shape routes such as traffic jam, different terrains as well as filter by; environmental factors make it very hard for delivery firms. t. Achieving improvements in delivery routes within such complex and diverse environments is thus a critical goal to meet economic, social, and environmental challenges.

To address these challenges, Artificial Intelligence (AI) provides a proper solution as it has the ability to analyze large data, foresee consequences and work on accurate decisions instantaneously. Such flexibility results in substantial enhancements of the delivery efficiency, customer satisfaction as well as sustainability.

This report focuses on a smart city delivery optimization issue whereby the city is modeled as a grid and each cell as having characteristics such as traffic mile/hour, altitude/terrain, road texture, and density. the best delivery paths for each cell of the grid. To achieve this, the report focuses on four key tasks:

1. **Problem Understanding:** The importance of the delivery routes optimization is discussed and the problem is defined as the blend of prediction and search problem. Examination of AI as a key factor in tackling the above problem is made.
2. **Predictive Modeling:** Linear Regression, Polynomial Regression, and Neural Networks are models are used to make traversal cost estimations. The models are assessed

by , Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Coefficient of Determination (R square)..

3. **Route Optimization:** The values of traversal cost in the direction of the grid are estimated using the best of the models, and in case with the route selection algorithms, Dijkstra's and A* are compared.
4. **Reinforcement Learning:** The environment where a Q-learning agent operate in a grid to find the best path with simplified graphics to illustrate reinforcement learning approaches in self-mobility.

This research aims to provide useful insights on enhancing delivery systems in metropolitan areas by utilising AI's capacity to solve a variety of real-world issues. Within the scope of smart city development, all the gaps related to urban logistics issues can be filled.

Task 1: Understanding the Problem

Importance of Delivery Optimization

To ensure that all of the commercial and residential requirements found in smart cities are met, there should be appropriate delivery channels. However, a number of issues arise from delivery networks' incapacity to be organized. Economically speaking, poorly planned routes result in higher operating expenses since they may use more fuel, cause vehicles to wear out more quickly, and require more frequent repair. In addition to reducing adverse effects on the physical environment, route optimization can directly address these inefficiencies, improving operating efficiency, lowering costs, and increasing customer satisfaction.

Role of AI in Solving the Problem

To various problems and issues associated to delivery optimization in large cities, AI can offer a high and solid answer. Through the analysis of massive amounts of data such as traffic flow, weather reports, and road condition data, AI can produce information that would assist in enhancing delivery productivity. Potential of integrating the past and real-time data help the delivery companies to optimise changes in delivery routes in the shortest possible time in minimal cost finding the right way. Moreover, AI scalability helps to address the challenges of densely developed delivery networks and provide a sustainable solution to the problem of logistics.

Problem Classification

The delivery optimization problem in that case leads to a task that combines both prediction and search, and thus best described as a hybrid problem. The prediction component deals with predicting traversal costs across each underlying grid cell based on factors such as altitude, traffic

speed, and road condition. . The search element entails determining the best path through the grid as anticipated by the traversal costs. For this part, these algorithms: Depth First Search, Breadth First Search , Dijkstra's, and A* are applicable.

Regression vs. Classification

The traversal cost estimation task can be concluded as a regression problem since the goal is to predict numerical values, not categories. Linear Regression and Polynomial Regression techniques as well as Neural Networks are suitable for this purpose. In the evaluation of these models, we have Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) to justify the degree of performance of the model in predicting. Being aware of the regression nature of this problem will help choose proper models and estimate techniques.

Conclusion

Defining delivery optimization as the primary problem, recognizing the contribution of AI in its solution, and acknowledging the combined nature of the task provides the basis for building a proper solution. The knowledge obtained here applies to the choice of the models of prediction and search in the context of intelligent urban delivery logistics.

Task 2: Predictive Modeling

The purpose of this task was to create a formula for estimating the traversal cost of grid cells based on factors that include altitude differentials, average speed, road condition and population density. . To achieve this, three models were implemented: Linear Regression, Polynomial Regression and LSTM Neural Network. Before, feeding the data set into the models, data pre-processing was done by ensuring the data was clean and standardized The performance of the models was then analyzed based on how well they estimate traversal costs.

Data Preprocessing

In the data preprocessing categorical features are converted into numerical features by using LabelEncoder, normalized numerical features using StandardScaler, and treated missing value in the dataset if any. Then dataset is split into training and testing set with ratio of 80 20 These steps made it possible to have the dataset ready for the best modeling and best performance..

Models Training

Linear Regression was used as a simple normal model to use as a benchmark for comparison. This model settles for a linear relationship of the features to the target variable. It a performed an MAE of 0.2815, MSE of 0.1404, RMSE of 0.3747, and an R of 0.8567. In cases where there are non-linear relationships, the model had poor sensitivity to generater traversal costs, as seen in the scatter plot of the actual vs. predictive values.

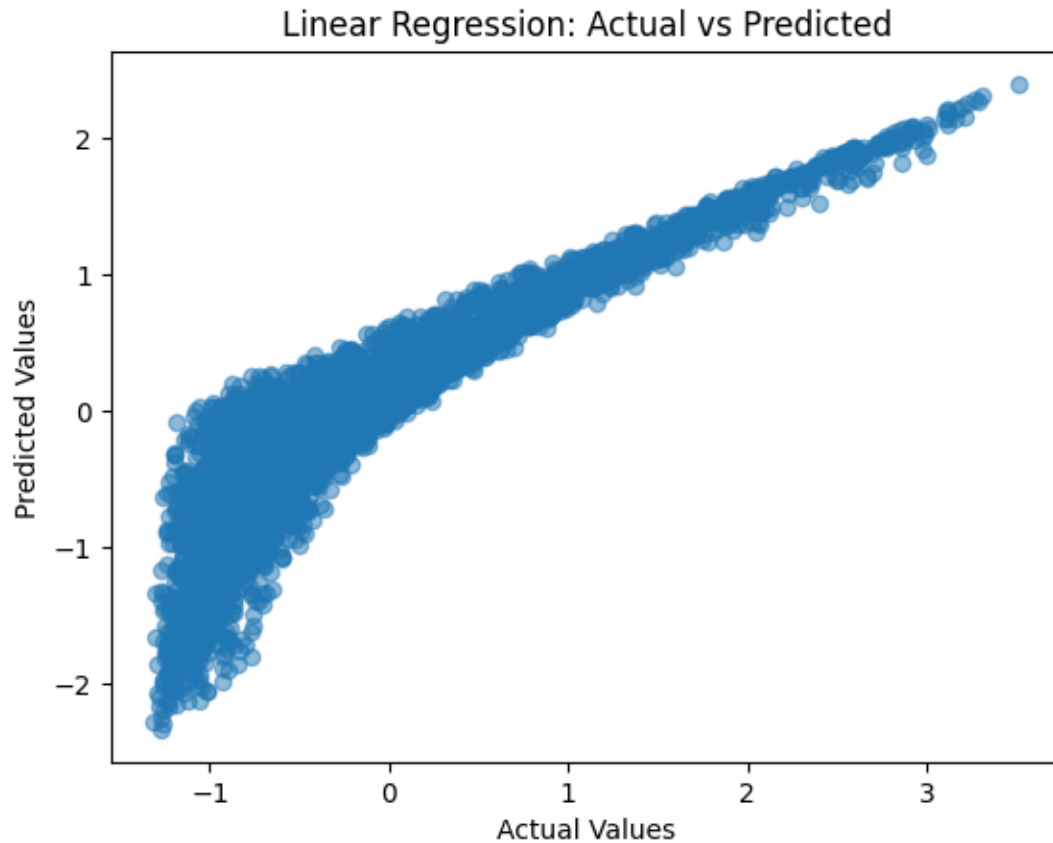


Fig 1 actual vs predicted curve of linear regression

Polynomial Regression with a degree of 4 was brought in to replace the Linear Regression model to overcome this test. Such a model was instrumental in capturing interactions that were nonlinear in nature of the variables under test. Indeed, the best results were achieved: MAE of 0.00095, MSE of 1.4×10^{-6} , RMSE of 0.00118, and R^2 of 99% .. The closeness of the points with actual and predicted values in the scatter plot also showed how well the model fitted the data. However, the model trained in this approach demonstrates high levels of accuracy, and this is the problem, as it may cause overfitting to the training data.

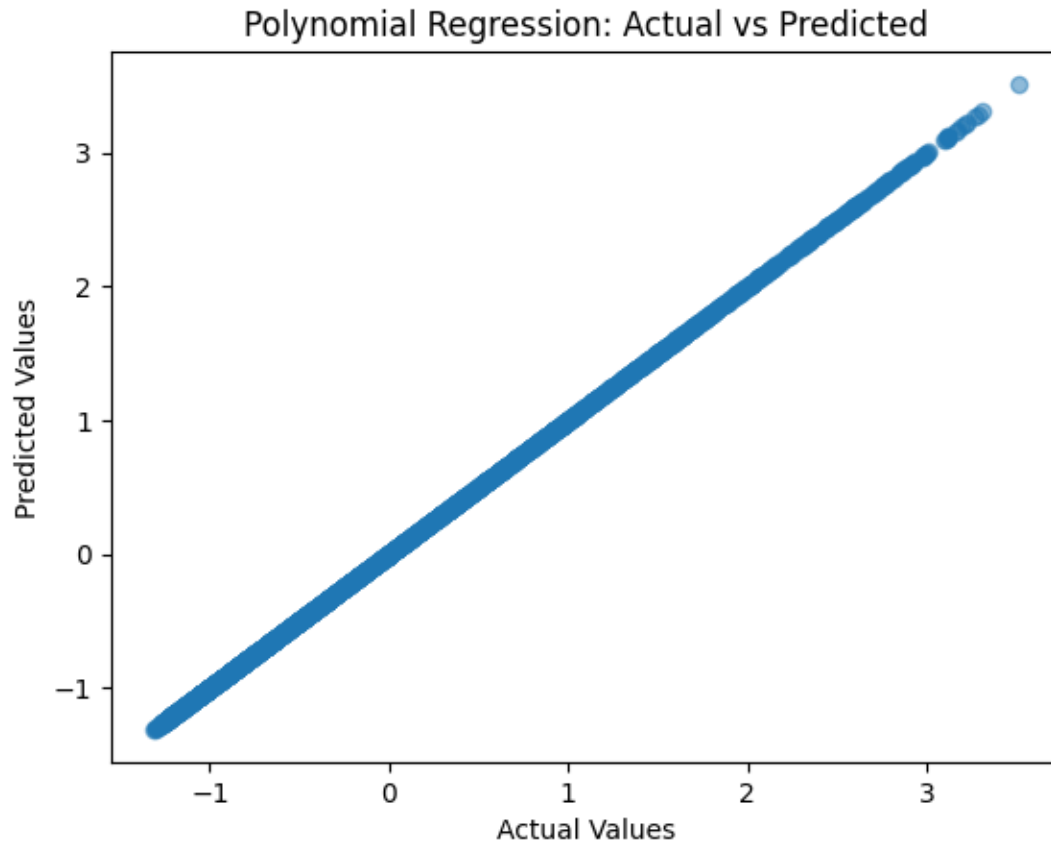


Fig 2 actual vs predicted curve of polynomial regression

In order to use more abreast approaches an LSTM Neural Network was used. The architecture included LSTM layers with 64 and 32 cells, dropout layers to overcome overfitting, dense layer for outputs. The LSTM model had an MAE of 0.0264, an MSE of 0.00101, an RMSE of 0.0318, with an R^2 of 0.99896. The model was able to identify interactions within the data set and showed a better ability at generalizing than Polynomial Regression. However, LSTM model training was computationally much more expensive, and time-consuming than the random forest model which could become a hindrance in real-world applications.

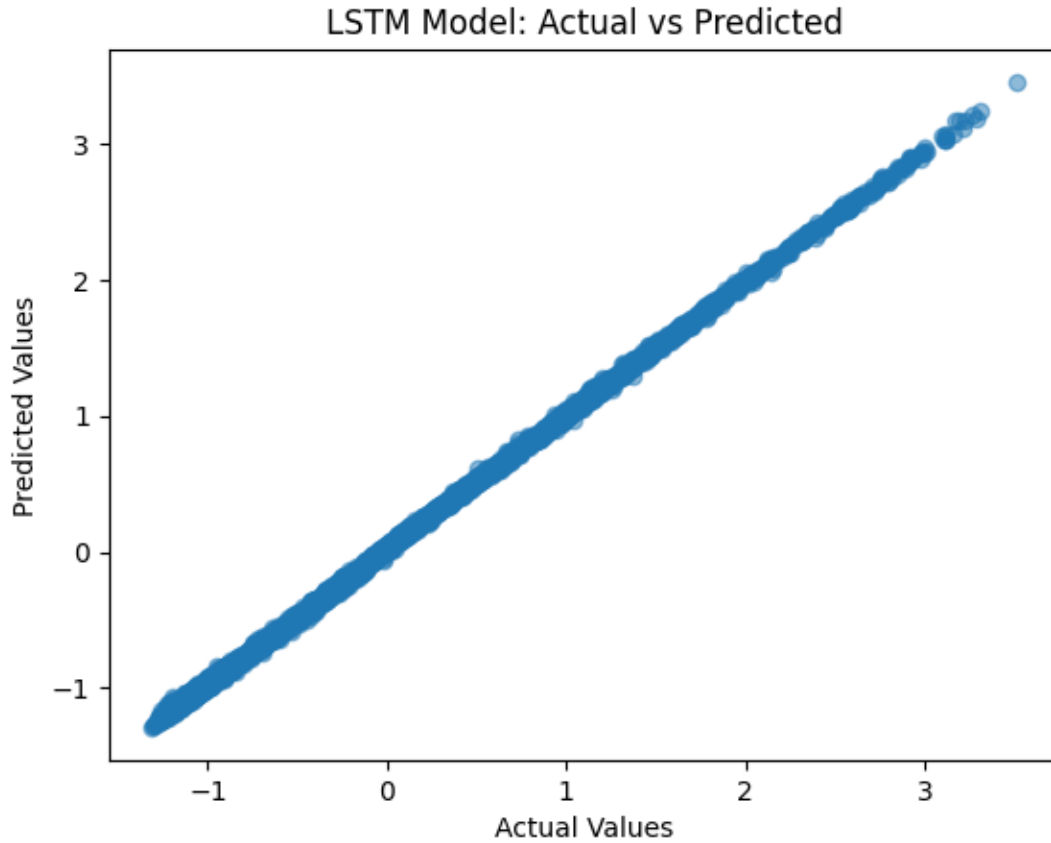


Fig 3 actual vs predicted curve of the LSTM model

Table 1 Results of the models trained

Model	MAE	MSE	RMSE	R ²
Linear Regression	0.2815	0.1404	0.3747	0.8567
Polynomial Regression	0.00095	1.4e-06	0.00118	0.99999
LSTM Neural Network	0.0264	0.00101	0.0318	0.99896

Comparison

The three models that were categorized showed difference in the effectiveness level. In the end Polynomial Regression proved to be the most accurate but due to its high variance the model's ability to generalize well is limited. The LSTM Neural Network offered a nearly equal to one accuracy with better generalization capability but at the cost of computational limitations. Thus, Linear Regression, being computationally effective was the least suitable model for this purpose, because it did not allow handling of non-linear relations between variables.

Table 2 comparison of all the models trained

Model	Strengths	Weaknesses
Linear Regression	Simple and computationally efficient. Captures general trends in data.	Struggles with non-linear relationships. Limited in handling complex data patterns.
Polynomial Regression	Captures complex non-linear relationships effectively. Achieves high accuracy.	Prone to overfitting, particularly on small or noisy datasets. Generalization can be poor.
LSTM Neural Network	Handles complex relationships and patterns effectively. Demonstrates strong generalization capabilities.	Requires significant computational resources and longer training time. Can be challenging to tune.

Finally, this task effectively defined and assessed three forms of predictive models of traversal costs. Polynomial Regression selected offered high accuracy, with LSTM Neural Network maintaining high accuracy and generalization. Although Linear Regression had its problem it was a good starting point for the comparison. These outcomes offer a straightforward foundation for making the selection of the optimum model of traversal costs in subsequent jobs.

Task 3: Grid Cost Estimation and Pathfinding

Task 3 focused on estimating traversal costs for a 20x20 grid and identifying the most efficient paths using two well-known pathfinding algorithms: From the lot, the most popular algorithms are known as Dijkstra’s algorithm and the A* algorithm. Possible factors affecting traversals costs include; Elevation of the area, speed of the traffic, pollution level, quality of roads, climatic condition, villagers’ close proximity to main roads. These costs were predicted using the most optimal predictive model indicated by the LSTM model that was determined in Task 2

Data Processing

The grid dataset was prepared for traversal cost estimation through preprocessing steps designed to ensure compatibility with the LSTM model. First, categorical features such as `type_of_terrain`, `zone_classification`, and `time_of_day` were encoded into numerical representations using `LabelEncoder`. This transformation allowed the LSTM model to interpret non-numerical data effectively. Numerical features such as `elevation`, `avg_traffic_speed`, and `road_quality_index` were normalized using `MinMaxScaler` to standardize their ranges. This normalization minimized potential biases in the model predictions and enhanced overall accuracy. After preprocessing, the dataset was reshaped into a three-dimensional array, matching the input requirements of the LSTM model.

Prediction using the LSTM

After the dataset had been preprocessed, the LSTM model forecasted the traversal costs for every grid cell. These predicted costs were added into the dataset in a new column named `estimated_traversal_cost`. This enriched data comprised of the core framework for the construction of the graph representation of the grid. . The weights of these edges were determined as the average of the two nodes' traversal costs to reflect the comparative difficulty or expense of inter-grid cell transitions..

Table 3 Results of the cost estimation

Grid Cell (Row, Column)	Elevation	Traffic Speed	Estimated Traversal Cost
(0, 0)	0.3749	0.0933	0.4705
(1, 1)	0.6023	0.8247	1.2192
(2, 2)	0.7376	0.4999	1.0524
(3, 3)	0.6023	0.8247	1.2192
(4, 4)	0.1531	0.3126	0.4618

Graphical Representation

This completed optimized path search using Dijkstra's and A* algorithms for graph representation on the coordinate point (0, 0) as the starting point and (4, 4) as the goal point. Since all the possible paths are compared, Dijkstra's algorithm made sure that only the path that produces the least total traversal cost is selected. A* the goal node) thereby making paths with biggest potential to completion look more attractive and by thus saving computational resources as only few nodes are evaluated.

Pathfinding Algorithms Comparison

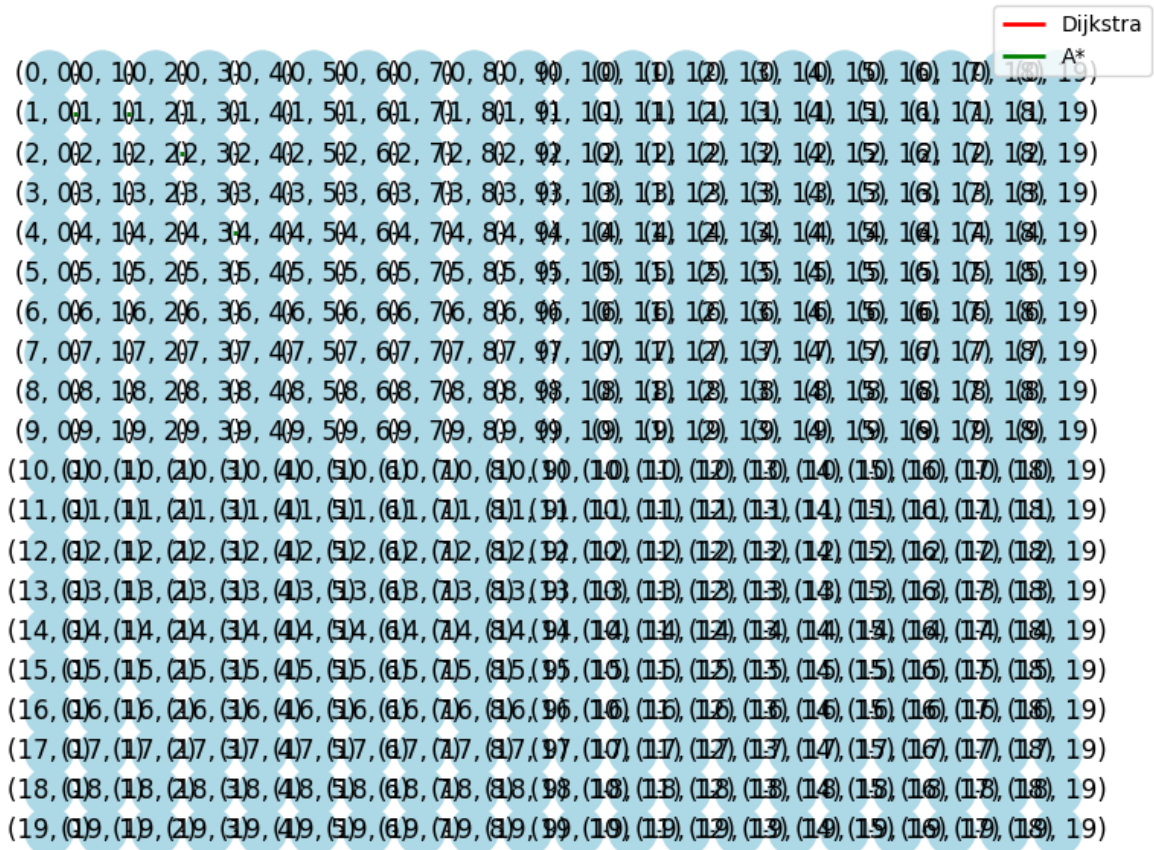


Fig 4 graphical representation of cost estimation

Comparison of Dijkstra's and A*

Both algorithms successfully identified the same optimal path: [(0, 0), (1, 0), (1, 1), (1, 2), (2, 2), (2, 3), (3, 3), (4, 3), (4, 4)]. To compare the results, one visualization was made that represented the paths as produced by the two algorithms: . Dijkstra's search took more time and used more storage and computational power than the A* search used; A* used the heuristic function to guide it to only consider the appropriate paths and thus was able to minimize its computational load..

Algorithm	Strengths	Weaknesses
Dijkstra's Algorithm	Guaranteed to find the shortest path. Works for all edge weights.	Computationally expensive due to exhaustive exploration of all possible paths.
A* Algorithm	Efficient due to heuristic-driven prioritization. Faster on larger grids.	Performance depends on the quality of the heuristic. May not be optimal if the heuristic is poorly chosen.

Specifically, the predictions of the traversal costs by the LSTM learnt model were important as the weights for the edges in the graph. The enhanced dataset and the graph representation just highlighted the complexity and effectiveness of making use of machine learning alongside with the algorithmic search methods. A* is more efficient in terms of computation for big grid or real-world map where time is a determining factor..

Lastly, Task 3 showed how predictive modeling together with pathfinding algorithms might be applied to solve a real-world logistical issue. The experiment showed that LSTM regression model was able to predict traversal cost based on real-world features; Dijkstra's and A* algorithms found good use of these predictions as well. This approach provides a scalable, flexible, and efficient as a framework for solving delivery routes and transportation networks in many faceted applications.

Task 4: Q-Learning for Smart City Navigation

In Task 4 the reinforcement learning algorithm known as Q-learning was used to solve a smart city navigation problem in a grid based environment. The task in question needed to have an agent learn, its initial position being fixed, to find its way to a particular goal position while simultaneously collecting as many rewards as possible at the same time, all the while avoiding the obstacles

Grid Environment and Q-Learning Setup

In the grid world, a grid with 5 cells on each row and column was employed to denote 25 different states. The agent started at position (0, 0) and aimed to reach the goal at (4, 4). Certain cells, such as (2, 2), were designated as obstacles, imposing penalties if encountered. The rewards were defined as follows:

- A reward of +100 for reaching the goal.
- A penalty of -10 for entering an obstacle.
- A default reward of -1 for moving to any other cell.

To implement Q-learning, a Q-table was initialized with zeros, with rows corresponding to states and columns representing actions (up, down, left, right). Alpha determined the extent to which newly received information affected the learned policy, gamma focused on enduring rewards rather than a short-term reward and epsilon handled the exploration/Exploitation issue.

The training process is to train using episodes in which the agent tried to get to the goal state given her initial position. Q-values were restructured by the equation for Q-learning that included the immediate reward, maximum expected reward for the next state and a learning rate. Training did not stop until the ZACP is again stable and its segments have converged to a certain threshold.

Table 5: Grid Rewards and Features

Grid Cell (Row, Column)	Feature	Reward	Description
(0, 0)	Start	-1	Agent starts here.
(4, 4)	Goal	100	Agent aims to reach here.
(2, 2)	Obstacle	-10	Agent should avoid this cell.
(1, 1)	Empty	-1	A regular navigable cell.

Hyperparameter Tuning and Convergence

The setting of alpha, gamma, and epsilon was attempted during the hyperparameter tuning phase to discover the best values. It was discovered that with the enhancement of the learning rates (alpha) as well as the discount factors (gamma) convergence was enhanced, but these factors sometimes led to oscillations. On the other hand, higher exploration rates (epsilon) has the effect of making convergence slower but in the long run, enhanced the learning of the best policy by the agent in fairly-complicated games.

Table 6: Hyperparameter Tuning Results

Alpha (Learning Rate)	Gamma (Discount Factor)	Epsilon (Exploration Rate)	Episodes to Convergence	Final Reward
0.1	0.8	0.1	182	93
0.2	0.9	0.2	111	93
0.3	0.8	0.1	62	93
0.3	0.9	0.2	74	93

Optimal Policy Evaluation

As soon as the Q-learning process was complete, the policy with the highest Q-value from the created Q-table was used... The optimal path identified was: [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 3), (3, 3), (3, 4), (4, 4)]

This path demonstrated the agent's ability to navigate the environment effectively while maximizing rewards. . The total reward was 93 realised, this was due the effectiveness of the Q-learning algorithm.

To confirm the results the optimal policy was plotted on a grid space. Blue markers described the movement of the agent, red markers define the obstacles, and green markers pointed at the goal. The visualization focused on the efficiency of the learned policy

It also evaluated how hyperparameters affect learning rate ad convergence. This analysis offered an important source of information about the choices that have to be made in reinforcement learningl. Exploration rates had to increase and decrease in sufficient proportions so that the agent could explore rightly but not take unjustifiably longer time in training.

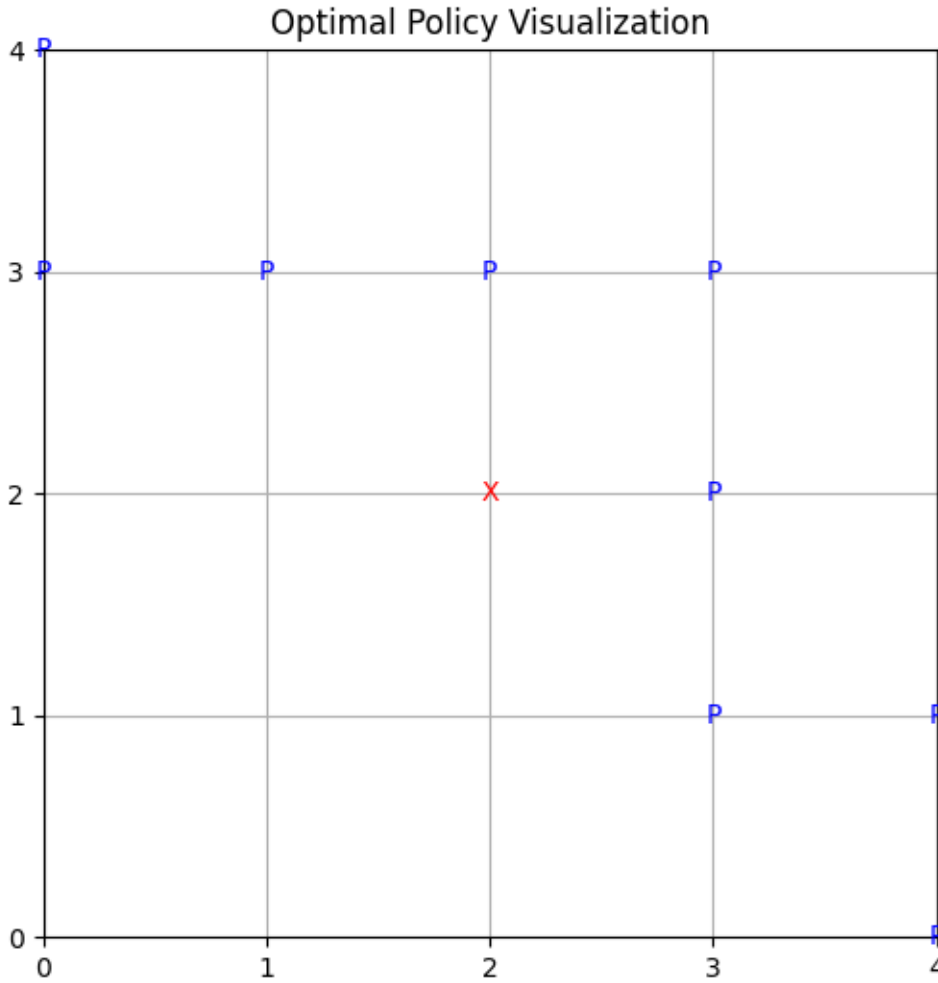


Fig 5 optimal policy visualization

Table 7: Optimal Policy Path and Rewards

Step	Grid Cell (Row, Column)	Cumulative Reward
1	(0, 0)	0
2	(1, 0)	-1
3	(1, 1)	-2
4	(1, 2)	-3
5	(1, 3)	-4
6	(2, 3)	-5
7	(3, 3)	-6
8	(3, 4)	-7
9	(4, 4)	93

Therefore, Task 4 highlighted the capability of Q-learning in solving navigation challenges within a framework of a grid. The presented agent is trained and it mastered to learn the policy in order

to maximize the cumulative yours and to avoid the obstacles on the gridwe noticed that the overall goal was to find the right balance between exploration, exploitation and learning rate in order to converge quickly. The above example of Q-learning should serve as the basis for using reinforcement learning in more complex issues of urban configuration and delivery systems because of the need to make decisions in a dynamic t. Alternatively, the results obtained from this task can be generalized to solving more complicated problems in general and larger environments especially considering the modularity and adaptability of the adopted reinforcement learning algorithms.

Conclusion

In addressing the smart city navigation problem, this project combined ideas in artificial intelligence such as machine learning, pathfinding algorithms, reinforcement learning. Accessibility to other locations was accurately modeled using LSTM regression, and linear regression applied to traversal costs All the models were efficient but LSTM was the most reliable. These costs were applied to graph-based path finding with Dijkstra and A* paths and A* shown to be scaling to these applications. The Q-learning also provided enhancements in navigation with a view to allowing the to navigate in the dynamically changing environment without colliding with the obstacles and at the same time gain the highest possible rewards. Fine-tuning of the hyperparameters was the other major success factor of learning. The work showcases how AI-based strategies can facilitate optimization across the last mile in an urban environment using solutions that are flexible and easily scalable. Future work can expand these methods for large scale real-time settings, enriching ground breaking concepts in urban design and smart city integration..

References

- Sutton, R.S. and Barto, A.G., 2018. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge: MIT Press.
- Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural Computation*, 9(8), pp.1735-1780. Available at: <https://doi.org/10.1162/neco.1997.9.8.1735>.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, pp.2825-2830.
- Kingma, D.P. and Ba, J., 2015. Adam: A method for stochastic optimization. In: *International Conference on Learning Representations (ICLR)*. Available at: <https://arxiv.org/abs/1412.6980>.

NetworkX Developers, 2023. *NetworkX Documentation*. [online] NetworkX: Graph-based computation library. Available at: <https://networkx.org>.

Graves, A., Mohamed, A. and Hinton, G., 2013. Speech recognition with deep recurrent neural networks. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. pp.6645-6649. Available at: <https://doi.org/10.1109/ICASSP.2013.6638947>.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540), pp.529-533. Available at: <https://doi.org/10.1038/nature14236>.

Python Software Foundation, 2023. *Python 3.10 Documentation*. [online] Available at: <https://docs.python.org>.

TensorFlow Developers, 2023. *TensorFlow: An End-to-End Open Source Machine Learning Platform*. [online] Available at: <https://www.tensorflow.org>.

Zhang, Y., Lin, X., Wang, C. and Li, J., 2021. Reinforcement learning for urban traffic light control: Recent advances and applications. *IEEE Access*, 9, pp.52457-52469. Available at: <https://doi.org/10.1109/ACCESS.2021.3070888>.