SMARTCOP – AN AUTOMATED PLATFORM TO MITIGATE THE IMPACT OF ROAD ACCIDENTS

2020 - 052

Sewwandi A.K.T

(IT16165830)

BSc (Hons) in Information Technology
Specializing in Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

September 2020

SMARTCOP – AN AUTOMATED PLATFORM TO MITIGATE THE IMPACT OF ROAD ACCIDENTS

2020 – 052

Sewwandi A.K.T.

(IT16165830)

Dissertation Submitted in Partial Fulfillment of the Requirements for the Bachelor of Science (Hons) in Information Technology Specializing in Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

September 2020

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: …………………………… Date: ……………….

[Sewwandi A.K.T]

The above candidate has carried out research for the bachelor’s degree dissertation under my supervision.

Signature of the supervisor: …………………………… Date: ……………….

[Dr. Windhya Rankothge]

Abstract

Since there is a considerable inclination of increasing road accidents the police administration finds it difficult to accurately schedule police officers to reduce traffic accidents. besides in most countries, a manual procedure is using to schedule traffic police officer which has many drawbacks. Even though several statistical applications are available there is still a necessity for an improved scheduling and enforcement platform of traffic police officers. As a result, we have come up with a solution called" SmartCop" which comprises an automated platform for traffic police officers scheduling to mitigate the impact of road accidents. The developed component is consisting two major sub components as, recommending traffic police officers based on their experience analysis, then schedule the recommended officers for different time zones according to the predicted results of road traffic accident prediction component. An unsupervised learning approach was used to build the recommendation system and a mathematical model was proposed and developed based on optimization to schedule recommended officers. The results show that the police officers were scheduled properly according to the requirement of each time zone predicted as the first component of the SmartCop application which predicts the road accidents.

Keywords: Traffic police officers schedule, Traffic police officers recommendation, Supervised learning, Optimization

ACKNOWLEDGEMENT

I would like to offer my heartiest and sincere gratitude to my supervisor Dr. Windhya Rankothge for providing perpetual guidance, advices, and motivation with her immense knowledge and patience throughout the research. Also, I would like to express my earnest gratitude to my co-supervisor Ms. Narmada Gamage for her continuous support and guidance. I extend my appreciation to the Sri Lanka Institute of Information Technology (SLIIT) for granting an opportunity for undergraduates to evolve with a valuable research project and my evaluation panel who have provided new directions pointing out mistakes. Especially the support and the supervision given by the team of academic experts in the Comprehensive Design and Analysis Project (CDAP) module with Dr. Janaka Wijekoon in the head chair is treasurable to make the research a success.

A special thanks offer to Mr. Laksiri Geethal who is the former Senior Superintendent of Police (SSP) in Mirihana Police Station, Sri Lanka, and the staff at Mirihana Police Station. Also, I would like to thank Mr. Amila Kothalawala, who is the Solution Architect at Cybergate Services (Pvt) Ltd for supporting us to contact Mr. Laksiri Geethal by providing further directions.

Finally. I would like to thank my family and friends for their constant support encouragement and valuable advices. Also, a special thanks offer to all the people who have helped even from a word to make the SmartCop research project a successful one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
WHO	World Health Organization
KNN	K Nearest Neighbors
IDE	Integrated Development Environment
ILP	Integer Linear Programming
BIP	Binary Integer Programming
OT	Over Time
REST	Representational State Transfer
API	Application Programming Interface
ML	Machine Learning
AI	Artificial Intelligence
GUI	Graphical User Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
RAM	Random Access Memory

LIST OF APPENDICES

1. INTRODUCTION

1.1 Background Literature

Road traffic fatalities and injuries have become one of the major global health problems according to WHO statistics. Every year over 1 million death and 20 to 50 million injuries were reported and more have found that road traffic injuries as a prime factor for fatalities worldwide, among people between 15-29 ages in years [2] and it was identified as a negative effect of the transportation system which needs a systematic solution to reduce road traffic accidents [3].

However different countries have come up with distinct solutions utilizing diverse technologies worldwide. A proper traffic police officers scheduling system is a greater solution as many countries in worldwide are trying different approaches to schedule and enforce traffic police officers at appropriate time and locations [3]. Nevertheless, even different countries follow different approaches to solve this matter, the problem of determining a proper working schedule in accordance with the needs and preferences of police stations still remains. Due to exponential growth of economic construction and transportation, the need of a proper police officers scheduling system has become a leading requirement. Problems such as, "How to strengthen the management of traffic police? How to complete traffic management work efficiently? How to make full use of efficient resources?" are major matters that needed to be resolved by the Government of many countries [5]. Most of the existing scheduling systems are usually following a manual approach by commanders of police stations which is inconvenient and time consuming [4].

Why manual staff scheduling is not a good solution?

Manual staff scheduling process is having many drawbacks due to staff volume, job complexity and also the unpredictability of the future tasks going to be allocated for officers in a police station. For an example a sudden change of the job priority due to an unexpected situation that could happen in a small area, or an incident that could impact on whole country or even for the whole world, during the valid period of manually generated schedule, leads to huge inconvenient circumstances as the responsible people need to regenerate a schedule according to the requirements of newly occurred situation. The origin of Covid-19, the infectious disease caused by the most lately discovered Coronavirus, and the job priority of police officers' modification due to the new situation is a great example that officers have encountered recently.

Furthermore, as the situation and the tasks needed to be fulfilled changes, decision making process regarding the enforcement of the best fitting officers for relevant tasks in different locations gets more difficult and complex in a manual approach. And the worst thing is that the officers who are responsible for generating daily, weekly, or monthly schedules might have a little or no time to do other tasks except scheduling [6].

Moreover, if the responsible officers for generating the schedule changes or a new officer allocated for the scheduling task, he will take a considerable time to understand the environment and carry out the job effectively, which is very lengthy and costly in manual scheduling. Since the decisions that made by former officers are not recorded in any system, knowledge sharing about the scheduling process is going to be rely on few officers [7].

What makes it different in automated scheduling?

Maintaining a software to automate the officers scheduling process, will help to optimize the workforce and minimize the mistakes and inaccuracy occurring in manual scheduling. Officers scheduling software can take the responsibility to generate and manage the schedules according to pre-determined set of criteria without human interaction. These kinds of software can be integrated with forecasting applications implemented using AI technologies to accurately predict the most suitable set of officers or employees for the relevant task by analyzing different factors [8].

Not as time consuming, and difficult as in manual scheduling, the scheduling software can perform the same task but with more accuracy and fast. Also, an automated software capable of doing instant alternations due to condition changes and it will create time for senior staff who responsible for scheduling and enforcement to devote more time to what they actually need to do.

A better schedule with sufficient and appropriate duties, and officers' satisfied shift allocation, which includes fair division among officers will lead to a quality service which eventually lead to reduce road traffic accidents and injuries.

What others have done to develop traffic police officers scheduling?

The optimal allocation of police officers by the division and organization of labor has determined from the study conducted by Dragana Todovic et al. Solution was designed based on identified predefined scheduling patterns, needs of the police station and preferences of policemen. Required number of officers per shift, rules of shift work, number of working hours of police officers, the monthly working standard has considered as parameters to identify the most suitable shift patterns. A mathematical model has been implemented to minimize the required number of police officers per shift using mixed integer programming [4].

A study was conducted to explore the experience of police officers including the role of traffic police against arriving at the crash scene, the challenges they face, and their opinions about how to enhance care to road traffic injuries victims. Qualitative research was conducted through interviews and group discussions in order to obtain data. The results suggested to provide training and equipping officers so that they will be capable of delivering post-crash care on victims while emergency medical services are yet to be established [2].

An optimization-based decision support system for deploying patrol officers has been implemented. The system predicts the hourly needed officers to maximize coverage and allows them to meet human needs. And also, the system is capable of evaluating schedule changes and suggest alternatives. According to results, the implementation of police patrol scheduling system improved productivity by $11 million per year. Response time has been decreased by approximately 20 percent and officer morale was positive [9].

A mathematical model and solution algorithms were implemented by the authors of paper [10] as a solution for the problem of staff scheduling. Reducing the number of employees needed for daily assignments was their main objective. In order to solve the problem, they have separated the problem into two components respectively "sequence of working and rest periods for each employee" and "the daily assignment to be performed in each working period by each employee." ILP models and number of algorithms were used to formulate the first component while the second component was solved utilizing a heuristic approach. Results were able to determine an optimal solution for the employees worked within 6 months of period.

An algorithm has been proposed for optimized scheduling which increases employee satisfaction. The targeted workforce has been scheduled into four-hour blocks by analyzing 42 discrete shifts for a seven-day work week. By considering the characteristics of decision variables a BIP model has used to design and implement the model. In order to make it easier for employers to manage employees remotely, the model has been implemented as web platform [11].

A study conducted by the authors of the paper [12] have solved a problem of employee scheduling utilizing a hybrid Genetic algorithm with tournament selection and two-point crossover with three individuals. According to authors selection of three individuals instead of two individuals caused for generating better fitness value. Two problems have identified and concatenated with the repair operators respectively, choosing the penalty cost for the fitness function, and controlling the impact of the repair operators. And the results have determined an optimal time to begin with repair operators.

Two mathematical programming models have been developed respectively, an individual planning model and integrated planning model, and resolved it utilizing a mathematical programming software called “CPLEX”. The main objective behind the two models was to minimize the total officer patrol time. According to results authors have found that integrated planning model perform better than the individual planning model, and the routes were also efficiently organized in integrated planning model when comparing with the individual plan. Furthermore, the percentage of night time enforcement was decreased by 47% in integrated planning model [13].

1.2 Research Gap

In the above discussed papers authors have tried different optimization approaches to solve the staff scheduling problem. Most of the researches have Mathematical approaches while few of them chosen heuristic approaches. According to authors of the paper [10], mathematical approach gives more accurate results than the heuristic approach conducted by them. Different researches have selected distinct objectives to solve the scheduling problem. In some papers, authors have considered the employee satisfaction when implementing their objective function while some others have considered the number of officers per the shift and total patrol time when implementing the objective function.

Different authors have taken different time periods for their scheduling systems such as one month, one week, 6 months and also hourly schedules.

However, for better police officer’s enforcement it is not enough if we only concentrate on just scheduling officers, but we need to allocate relevant officer for the appropriate and right task by recommending matching officers for the matching task. If we do so we can come up with more accurate schedule rather than a schedule which tries to reach an optimal solution by considering one or two objective function. As depicts in table 1, above mentioned papers have not implemented a solution which include both recommendation and optimization to schedules officers or employees.

A comparison among SmartCop police officer recommendation and scheduling component and similar other existing applications has conducted as follows.

- Paper [2]: Traffic police officers’ experience of post-crash care to road traffic injury victims: a qualitative study in Tanzania
- Paper [4]: Police officer scheduling using goal programming
- Paper [9]: A Break from Tradition for the San Francisco Police: Patrol Officer Scheduling Using an Optimization-Based Decision Support System
- Paper [10]: Models and algorithms for a staff scheduling problem
- Paper [11]: Scheduling Algorithm with Optimization of Employee Satisfaction
- Paper [12]: Optimizing Employee Schedules by a Hybrid Genetic Algorithm
- Paper [13]: Optimal scheduling for police patrol duties

Table 1: Comparison among SmartCop traffic police scheduling component and existing works

Features	**Paper [2]**	**Paper [4]**	**Paper [9]**	**Paper [10]**	**Paper [11]**	**Paper [12]**	**Paper [13]**	**Smart Cop**
Recommending suitable officers for relevant task								✓
Use a mathematical approach to solve the scheduling problem		✓	✓	✓	✓		✓	✓
Use a Heuristic approach to solve the scheduling problem				✓		✓		
Use optimization to minimize the number of shifts allocation for each officer								✓
Use optimization to maximize the coverage and human needs			✓					
Use Optimization to minimize the required number of officers per shift		✓		✓				
Use optimization to increase employee satisfaction.					✓			
Use optimization to minimize total officer patrol time							✓	
Conducted an experience	✓							✓

analysis of officers								
Schedule duration (Hourly)			✓					
Schedule duration (6 months)				✓				
Schedule duration (1 week)					✓			✓
Schedule duration (1 month)		✓						

In the SmartCop platform, police officers' recommendation and scheduling component has considered the both officer recommendation and scheduling and came up with a better solution than the existing systems. The recommendation system was implemented based on analyzing officers' experience levels for particular reasons that caused the occurrence of predicted accidents. Then a mathematical model has been implemented using optimization with the objective of minimizing the number of shifts allocating for each officer within the schedule duration. The main intention behind the objective function by minimizing the number of shifts allocating for each officer is to avoid an officer exceeds the maximum number of shifts that an officer can have for a schedule period, otherwise, the extra shifts that an officer would allocate is going to be recognized as an overtime work which could cause an extra cost for the police station and it could be a hectic schedule for the allocated officer as well. Apart from that after generating an accurate schedule, a notification system was implemented to notify allocated officers.

1.3 Research Problem

Accurate traffic police officer scheduling is a complex problem in locally as well as globally. Poor scheduling of officers does not make any sense of reducing the rate of traffic accidents. This scheduling process usually performed manually in most countries all over the world including Sri Lanka [4]. It has shown that when the number of working hours of an officer increases, the risk, and severity of accidents increase exponentially [14]. And also if allocated number of shifts exceeds the maximum number of shift an officer could have within the valid period of the schedule it cause an extra cost as well as a hectic schedule for officers. Inappropriate shift schedule negatively affects the ability of police officers, their health, and quality of decision making [4]. It has found that traffic police officers are the first responders on the crash scenes in most situations and act as both rescuers of the victim as well as the traffic police officer [2]. Officers with low-level qualifications are challenged by their limited experience which creates a difficult environment that puts victims' lives at risk [2]. Therefore, a proper scheduling system for traffic police enforcement is a vast requirement all over the world. Even though several statistical applications available, there is still a necessity for the development of an improved system for scheduling and enforcement of traffic police officers.

1.4 Research Objectives

1.4.1 Main objective

- To implement a web application which is capable of appropriately scheduling suitable police officers for the relevant task in a cost-effective way, and display scheduled details accurately and in a descriptive manner while notifying officers about the allocation

1.4.2 Specific objectives

- Filtering top most predicted reasons for occurring accidents in each time zone.
- Implement the Recommendation system
    - Handling the dataset.
    - Recommending traffic police officers based on their experience analysis for each time zone.
    - Displaying recommended officer details within each time zone in the web application.
- Implement the scheduling system
    - Design a mathematical model for scheduling recommended officers.
    - Implement the designed mathematical model using optimization.
    - Display scheduled details appropriately in the web application.
- Send notifications for the allocated officers to notify about their allocation.

2. METHODOLOGY

2.1 Methodology

As the solution for the observed problem discussed previously, we have developed an automated platform called "SmartCop" which comprises a module to properly schedule traffic police officers by considering different time zones predicted according to the percentage of occurring accidents due to different reasons by the first component of SmartCop road accident prediction. The scheduling module has been implemented with two major functionalities respectively, a recommendation system for recommending suitable traffic police officers based on their experience analysis, and then the recommended officers were scheduled to available checkpoints appropriately as the second component.

The police officer's dataset utilized for this analysis by an online data delivery source [1]. According to Fig. 1, the preprocessed dataset will be an input for the officers' recommendation system. As the recommendation and scheduling were done according to the overall prediction of the road accident, which is the first component of SmartCop platform, prediction results also taken as an input to the scheduling system. As a combined result of the frequency prediction and the reason prediction, the frequency of accidents for each reason at each time zone has been taken into consideration to perform the recommendation. Then the scheduling of recommended officers will perform as an optimization problem, and will take the results of recommended officers in each time zone as inputs. The results of both the recommender system as well as the scheduling system converted to json format document and stored in the database. Finally the overall results of SmartCop scheduling component will pass to the front end of the application which is a web application using REST APIs.

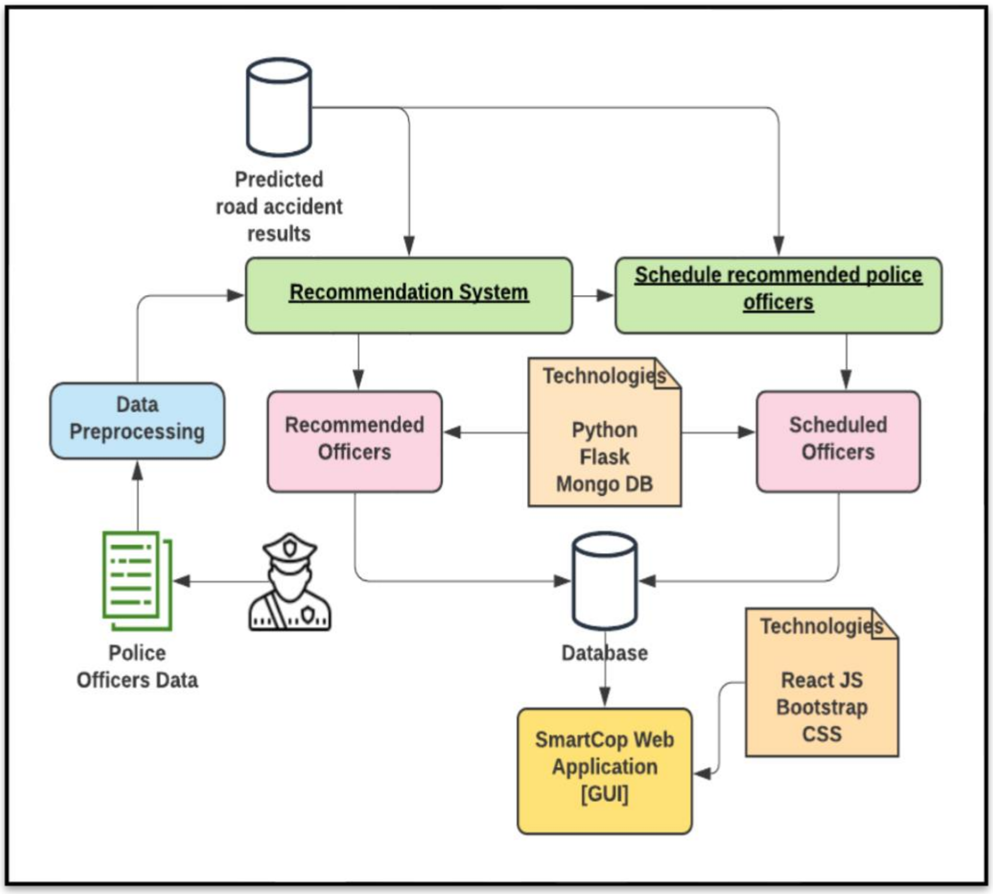


Figure 1: Road accident prediction system diagram

2.1.1 Feasibility study

The SmartCop web application was a result of a research for solution to mitigate the impact of road accidents. The second main component, traffic police officers recommending and scheduling was identifying as a major requirement of police worldwide. Even though several scheduling systems have implemented in different countries, an automated scheduling system with a recommendation system haven't implemented still which is would be a great solution for the problem of rapid increase in road traffic accidents.

Considering the huge demand of the application, and the resources, time and cost approximately allocated for the whole project, it was identified the development of the application is feasible. For the initial development of the application has not charged any cost, however, for adding additional features to improve the performance and hosting the application will be cost some amount which can be accepted as a considerable budget.

As the initial phase the application has developed for one police station, and the system can be expanded in the future according to the client' requirement. Furthermore, feasible deadlines is there through out the whole software lifecycle from requirement analyzing to testing the final application.

Moreover, the required data collection and other information gathering process was easy to handle and the knowledge and skills required for develop the application was manageable. In addition, all the required tools and technologies for the initial phase are freely available

2.1.2 Functions

- ➢ Register to the application: Sign Up after providing the requested details and create a user profile
- ➢ Sign In process
- ➢ Manipulate the created user profile
- ➢ View overall road accident prediction details as visualized summery
- ➢ View recommended police officer details including their experience for each time zone.
- ➢ View schedule details
- ➢ Send notifications for scheduled officers

2.1.3 Application literacy

The language used in the web application including the traffic police officer's recommendation and scheduling is English and because of that this application can be used as any user worldwide. Further, this application is going to be deliver in Tamil and Sinhala in the future as this product is going to be available in Sri Lanka in the first place. A proper internet connection is a major requirement since this product is a web application and it is important when fetching data from servers through APIs to maintain the stability and functionality of the application. Moreover, Authentication is one of the major security requirements a web application should have and SmartCop is having a sign-in and sign-up functionalities to ensure that the clients try to access the application are actual authenticated users.

2.1.4 Required resources

<u>Software requirements</u>

1. **PyCharm**

PyCharm is a cross-platform IDE which supports Windows, macOS, as well as Linux. It was mainly designed for Python developers as a JetBrains product. More than thousands plugging are there compatible with PyCharm and also developers can implement their own plugins using the API provides by PyCharm [15]. However, PyCharm has been used to develop the backend of the SmartCop project since the main language used to implement the backend was Python.

2. **Visual Studio Code (VS Code)**

Visual studio code is a freely available source code editor developed as a Microsoft product and available for Linux, windows as well as macOS. Int consists with lots of features that is more important for software developers [16]. Different languages and frameworks are supported by VS code while having quick and easy installation process. This code editor has been used to implement the front- end of the SmartCop web application using React.JS

3. **Python**

Python can be identified as one of the easiest to use and learn language with simple syntax and code written in Python can be executed much faster than other programming languages. Because of this simplicity and performances, Python has become a widely used language in ML and AI to implement complex algorithms. Especially, Python comprises extensive set of libraries for AI and machine learning such as, NumPy, Pandas, Seaborn, Kera’s, TensorFlow, Scikit-Learn and many more [17]. Since the backend of the SmartCop project basically based on ML Python has used to do the implementation.

4. Flask

Flask is classified as a micro web framework written in Python and it is considered more Pythonic than the Django we framework. Flask does not comprise a data abstraction layer or form validations which provides by pre- existing third-party libraries. Flask provide many important features for Python developers such as, RESTful requests dispatching, Support for secure cookies, development server and debugger, and many more [18]. In SmartCop Flask has used to implement RESTful APIs and develop the server-side in the police officer recommendation and scheduling.

5. MongoDB Atlas

MongoDB is an open-source cross platform, document oriented leading NoSQL database which is written in C++ [19]. Several different capabilities like, flexible data modeling, indexing and high-speed querying which are very useful in implementing ML algorithms are provided by MongoDB. MongoDB Atlas has been used as the database of the project which is a cloud-based database fully managed and developed by MongoDB

6. ReactJS

ReactJS is an open-source JavaScript Library utilized to implement reusable GUI components which is developed and maintained by Facebook. ReactJS abstracts away the DOM and offer the user as a simple programming model with better performance [20]. Hence SmartCop used ReactJS to implement the front-end of the web application.

7. HTML and CSS

HTML and CSS are the two core technologies used for building web pages. HTML is mainly using to develop the structure of a web page while CSS style the web page. They have been utilized to create documents in the World Wide Web relevant to the SmartCop web application.

8. **Bootstrap4**

Frontend UI design is supported by the Bootstrap4 which is the latest version. It is the most demanded HTML, CSS, and JavaScript framework to implement efficient and responsive websites. It is free to be utilized.

9. **Anaconda Navigator and Jupyter Notebook**

Anaconda Navigator is a desktop or GUI launched by the Anaconda distribution that facilitates to launch software solutions [21]. Also, instead of using commands in a command line, it flexibly handles the conda packages, channels, and environments. However, the navigator is available for both Windows, macOS, and Linux. In addition, Jupyter Notebook is a software application which can be navigated via the Anaconda Navigator and it has been utilized to implement and test road accident prediction models in SmartCop project.

<u>Hardware requirements</u>

- ➢ A server machine with high processing power for backend
- ➢ Any Personal Computer (PC) or laptop with proper internet connection

<u>Memory requirements</u>

Since the SmartCop web application needs ML and it has large amount of data, some higher memory is required for backend but that much memory is not required for the frontend.

Back-end (Python)

- ➢ RAM - 8 GB minimum, 16 GB or higher is recommended
- ➢ Data Storage – Maximum of 2GB

Front-end (ReactJS)

- RAM – 1GB - 2GB
- Data Storage – 500 MB or higher

2.2 Commercialization Aspects of the Product

With the help of a police officer's scheduling tool, you can build a perfect traffic police officers schedule in one click. most existing systems are performed as manual systems and separate officers are allocating for creating daily weekly or monthly schedules depending on the police department, area or the country. Because of that in every month, an extra cost has to be spent for the responsible party who are in charge of creating the schedules.

AS a solution SmartCop has implemented an automated platform to schedule traffic police officers weekly. The system does not need any human interaction to perform and it will automatic the duty of officers responsible for scheduling. Therefore, those officers also can focus and allocate their time and effort to the actual work they are supposed to do. And this will retrench the expenses allocated for scheduling and save officers' time.

If the generated schedule allocates officers with low level experience for relevant checkpoints, that would be not a proper schedule which might create a difficult environment that put victim's lives at risk. police officers may have different experiences regarding different types of accidents. In an accurate schedule experienced officers for different types of situations needed to be assigned so that it will lead to reduce the likelihood of occurring accidents in the relevant area. Considering that SmartCop has implemented a recommendation system by analyzing officers' experience and integrated it to the scheduling system. this will make sure all the checkpoints are enforced with suitable officers which will lead to reducing the impact of traffic accidents that can be happened in the future.

it will directly impact on reducing damages for common infrastructure, vehicles, physical elements which supports country's economy enormously. apart from all the mentioned benefits this will save human lives which is a major concern of society point of view and secure people who are capable of working for country and enhance the economy.

the main objective of SmartCop is optimally scheduling to minimize the number of shifts allocating for each officers and avoid the likelihood of an officer exceeds the maximum number of shifts that an officer can have within the valid schedule period. Otherwise extra shifts going to be worked by officers will be considered as an OT work which could cause an extra cost for the police station.

Furthermore, the platform facilitate,

- A recommendation system and scheduling officers in a single platform.
- End users to utilize the system with less technical knowledge.
- Recommendation and scheduling system can extend worldwide instead of restricting to one police station.

The SmartCop police officers scheduling platform is a web application which comprises two main components. as per the commercialization aspect the police officers scheduling functionality can be served as a free package. in order to get more accurate performances, client can be subscribed to premium version which comprises the police officers recommendation functionality.

Moreover, SmartCop web application notification system is another premium functionality that the client can subscribe to. Utilizing this feature police station can easily notify the relevant scheduled officers automatically. in the future new expanded features can be added to the premium version which will provide users to have more accurate performances

2.3 Testing & Implementation

2.3.1 Testing

Testing phase is one of the critical phases in the Software development Lifecycle and the time developers focus on investigation and discovery. This phase will help to ensure that the actual software product matches expected requirements of the client by executing manual or automated software texting processes. The main intention behind software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. It is very important to have a testing phase to every software developing process since bugs and errors can be identified early and can be solved before delivery of the software product. In this implementation, there are several testing phases such as design testing, unit testing, module testing, component and integration testing, and user acceptance testing. Since the traffic police officer's recommendation and scheduling is a sub part of the SmartCop web application, system testing is conducted after combining all sub parts of the application.

- Design testing
  - Design testing has been conducted during the design phase of police officers recommendation and scheduling component. It is tested whether the required functional requirements are included in the component design. Also, the required UI designs are tested whether they fulfill the best user experience. In addition, the flow of UIs, and user controls are tested during the design testing phase.

- Unit testing
  - Unit testing is conducted after implementing a piece of code unit or a module in the component and performed individually. Main goal of this

testing is to ensure whether the implemented piece of software unit is working properly. If there are bugs, they were fixed at the same phase.

- Module testing
    - In module testing, certain subcomponents of the road accident prediction component are tested. In addition, classes procedures, functionalities, and connection controls are tested in here. Main modules include the Police officers recommendation component and scheduling component.

- Component and integration testing
    - Whole police officer scheduling component is tested after integrating all subcomponents including the recommendation system and the scheduling component. If there are bugs found, they were fixed in the same testing phase. Especially, the integrated component's functionality is tested to ensure whether it is working as expected.

- User acceptance testing
    - In this testing phase, the whole component is tested whether it fulfills the business value of the product. Main goal is to satisfy the customer requirements and their acceptance level towards the working component.

Selected test cases have described below using Table I-IV

Table 2: Test Case 001

Test case ID	001
Test case scenario	Test for user sign in with valid credentials
Test steps	Enter valid email Enter valid password Click on submit button

Test data	Username = officer001@gmail.com
	Password = officer@123

Expected results	Sign-in to the application successfully
Actual results	Sign-in to the application successfully
Pass/ fail status	Pass
Test case ID	001
Test case scenario	Test for user sign in with valid credentials
Test steps	Enter valid email Enter valid password Click on submit button

Test data	Username = officer001@gmail.com
	Password = officer@123

Expected results	Sign-in to the application successfully
Actual results	Sign-in to the application successfully
Pass/ fail status	Pass

Table 3: Test Case 002

Test case ID	002
Test case scenario	Test for user sign in with invalid credentials
Test steps	Enter invalid email Enter invalid password Click on submit button

Test data	Username = officer001@gmail.com
	Password = officer@111

Expected results	Prompts the error message “Invalid Credentials”
Actual results	Prompts the error message “Invalid Credentials”
Pass/ fail status	Fail
Test case ID	002
Test case scenario	Test for user sign in with invalid credentials
Test steps	Enter invalid email Enter invalid password Click on submit button

Test data	Username = officer001@gmail.com
	Password = officer@111

Expected results	Prompts the error message “Invalid Credentials”
Actual results	Prompts the error message “Invalid Credentials”
Pass/ fail status	Fail

Table 4: Test Case 003

Test case ID	003
Test case scenario	Test for user View recommended officer details
Test steps	User navigates to police officer dashboard User clicked on the time zone he wants to search It will navigate to the officer recommendation page for relevant time zone User click on the "Display recommended officer details" button
Expected results	Display all the recommended officers
Actual results	Display all the recommended officers
Pass/ fail status	Pass
Test case ID	003
Test case scenario	User searches for a specific area to observe number of accidents in the area
Test steps	User navigates to home page Search for an area in the search bar Click on search icon
Test data	Region 202
Expected results	Displays number of predicted accidents in the searched area
Actual results	Displays number of predicted accidents in the searched area
Pass/ fail status	Pass

Table 5: Test Case 004

Test case ID	004
Test case scenario	Test for user View Scheduled officer details
Test steps	User navigates to police officer dashboard User clicked on “View schedule details” It will navigate to the officer schedule dashboard. User clicked on the time zone he wants to search It will navigate to the officer schedule page for relevant time zone User click on the “Display scheduled officer details” button
Expected results	Display all the scheduled officers
Actual results	Display all the scheduled officers
Pass/ fail status	Pass
Test case ID	004
Test case scenario	User searches for a non-existing area to observe number of accidents
Test steps	User navigates to home page Search for non-existing area in the search bar Click on search icon
Test data	Region 2
Expected results	Error message “Please select an existing area”

Actual results	Error message “Please select an existing area”
Pass/ fail status	Fail

2.3.2 Implementation

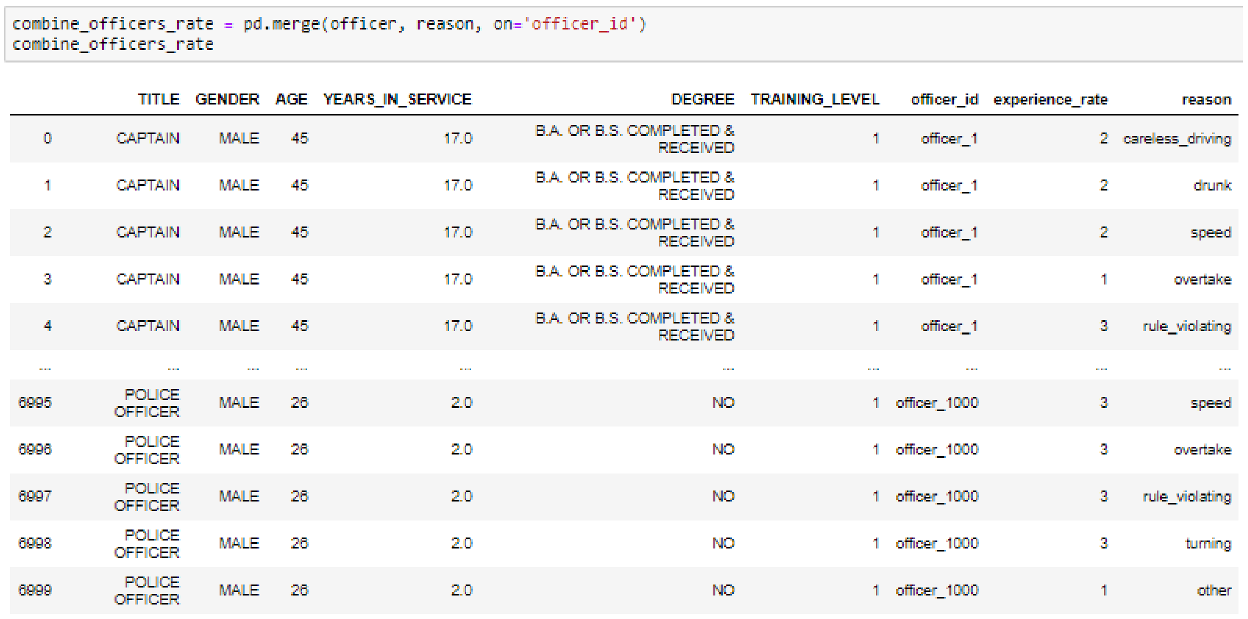
The traffic police officer’s recommendation and scheduling component has been implemented as the second component of the SmartCop web application. As previously mentioned, resources were used to develop the implementation of the whole module. The second component of the SmartCop was mainly implemented as two separate models.

- Model Implementation for the Police officer recommendation system.
  - The recommendation model was implemented using KNN algorithm with the Cosine Similarity.
  - Officers will be recommended for four time zones
- Model implementation for the police officer scheduling system.
  - Scheduling problem was identified as an optimization problem and designed and implemented a mathematical model using Gurobi Optimizer.

<u>Model implementation</u>

Data collection & Preprocessing

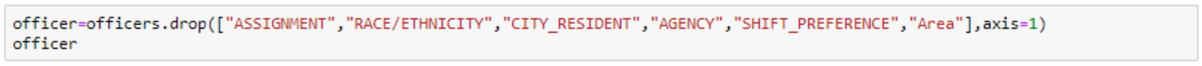
The dataset which has ben used for the recommendation and scheduling is available online [22] regarding traffic police officers generated 4 years ago. The collected dataset modified by merging another dataset which comprises the reasons which were the output of reason prediction of the SmartCop first component with experience rate of each officer for each reason as Fig. 2 depicts. And then less important fields or features were removed as follows in Fig. 3 and checked for null values. As depicts in Fig. 4 there we no null values found in the dataset. And also, the concise summery of the data frame as represent in Fig. 5 will also prove that there are no null values in the modified dataset.



```
combine_officers_rate = pd.merge(officer, reason, on='officer_id')
combine_officers_rate
```

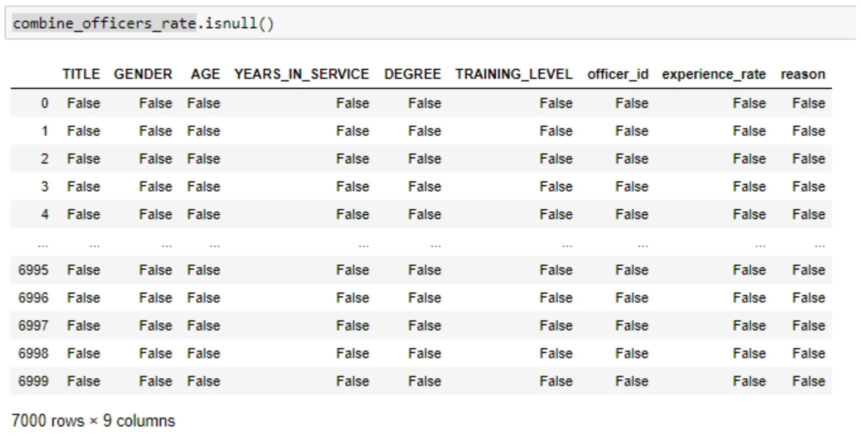
	TITLE	GENDER	AGE	YEARS_IN_SERVICE	DEGREE	TRAINING_LEVEL	officer_id	experience_rate	reason
0	CAPTAIN	MALE	45	17.0	B.A. OR B.S. COMPLETED & RECEIVED	1	officer_1	2	careless_driving
1	CAPTAIN	MALE	45	17.0	B.A. OR B.S. COMPLETED & RECEIVED	1	officer_1	2	drunk
2	CAPTAIN	MALE	45	17.0	B.A. OR B.S. COMPLETED & RECEIVED	1	officer_1	2	speed
3	CAPTAIN	MALE	45	17.0	B.A. OR B.S. COMPLETED & RECEIVED	1	officer_1	1	overtake
4	CAPTAIN	MALE	45	17.0	B.A. OR B.S. COMPLETED & RECEIVED	1	officer_1	3	rule_violating
...	...	...	...	...	...	...	...	...	...
6995	POLICE OFFICER	MALE	26	2.0	NO	1	officer_1000	3	speed
6996	POLICE OFFICER	MALE	26	2.0	NO	1	officer_1000	3	overtake
6997	POLICE OFFICER	MALE	26	2.0	NO	1	officer_1000	3	rule_violating
6998	POLICE OFFICER	MALE	26	2.0	NO	1	officer_1000	3	turning
6999	POLICE OFFICER	MALE	26	2.0	NO	1	officer_1000	1	other

Figure 2: Combined data frame of two data sets.



```
officer=officers.drop(["ASSIGNMENT","RACE/ETHNICITY","CITY_RESIDENT","AGENCY","SHIFT_PREFERENCE","Area"],axis=1)
officer
```

Fig.3: Removing less important features from the dataset

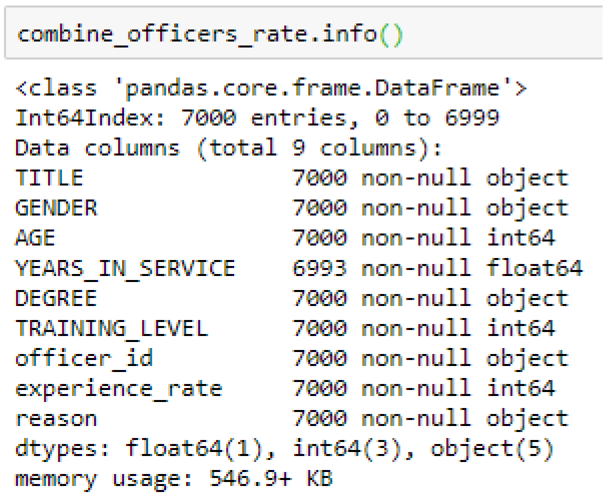


```
combine_officers_rate.isnull()
```

	TITLE	GENDER	AGE	YEARS_IN_SERVICE	DEGREE	TRAINING_LEVEL	officer_id	experience_rate	reason
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
6995	False	False	False	False	False	False	False	False	False
6996	False	False	False	False	False	False	False	False	False
6997	False	False	False	False	False	False	False	False	False
6998	False	False	False	False	False	False	False	False	False
6999	False	False	False	False	False	False	False	False	False

7000 rows × 9 columns

Fig. 4: checking for null values



```
combine_officers_rate.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7000 entries, 0 to 6999
Data columns (total 9 columns):
TITLE                7000 non-null object
GENDER               7000 non-null object
AGE                  7000 non-null int64
YEARS_IN_SERVICE     6993 non-null float64
DEGREE               7000 non-null object
TRAINING_LEVEL       7000 non-null int64
officer_id           7000 non-null object
experience_rate      7000 non-null int64
reason               7000 non-null object
dtypes: float64(1), int64(3), object(5)
memory usage: 546.9+ KB
```

Figure: 4: concise summery of the data frame

After the data collecting and preprocessing process, the main implementation will be started. That will be discussed in this report under two main sections.

- Model Implementation for the Police officer recommendation system.
- Model implementation for the police officer scheduling system.

Model implementation of police officer recommendation system

For the recommendation system, A collaborative filtering approach has been used to implement the model using KNN algorithm Since KNN is a very basic and common approach using for implementing the recommendation systems [23]. The basic idea behind the KNN algorithm is to find the most K number of users as the nearest neighbors to a given user. Along with the KNN algorithm Cosine Similarity has been used to measure the distance between data points by calculating difference of angle between two vectors as depicts in Fig. 6.

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

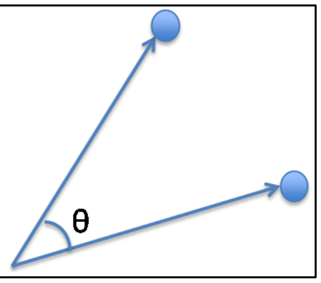
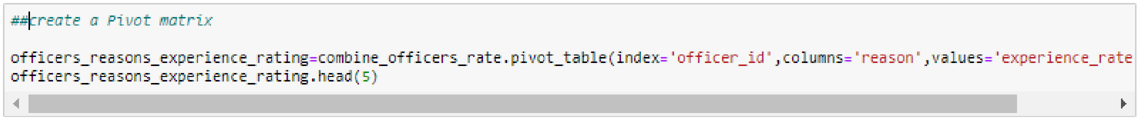


Figure 6: How Cosign Similarity works

Source: http://cs.carleton.edu/cs_comps/0910/netflixprize/final_results/knn/index.html

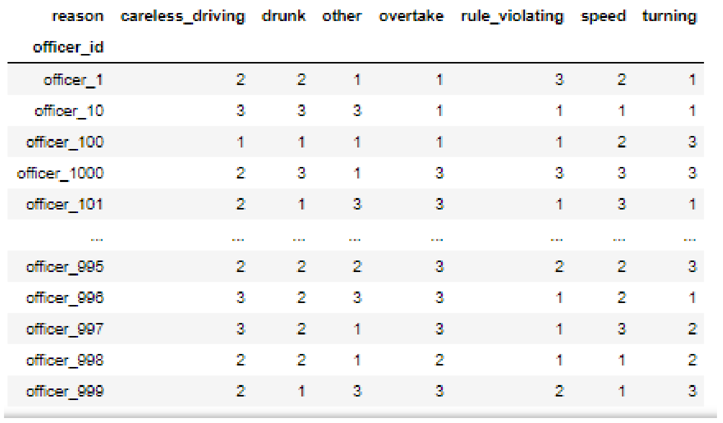
In order to do the calculation more efficiently the modified data frame was converted into a pivot table, as taking the index as ‘officer_id’, column as ‘reason’, and values as experience rate within the bellow code snippet in Fig. 7.



```
##create a Pivot matrix

officers_reasons_experience_rating=combine_officers_rate.pivot_table(index='officer_id',columns='reason',values='experience_rate
officers_reasons_experience_rating.head(5)
```

Figure 7: Creating the pivot table

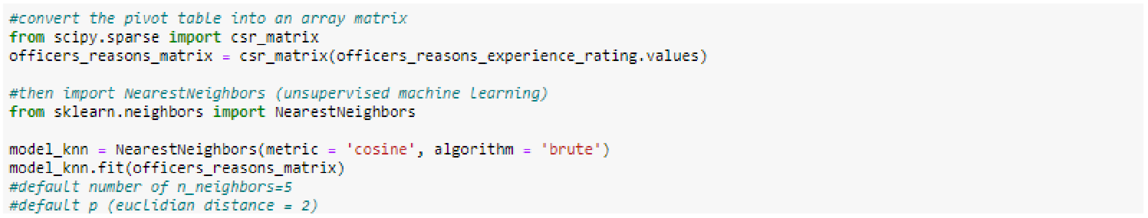
In the generate pivot table as represented by Fig. 8, the index are represented by officer ids and the columns or the features of the table are represented by the reasons for occurring accidents and the values under each column and index is the experience rate or experience level an officer is having under particular reasons. The values are rating from 1 to 3 where, 1 represent the low experience level while 2 and 3 represents respectively moderate and high experience levels.



reason officer_id	careless_driving	drunk	other	overtake	rule_violating	speed	turning
officer_1	2	2	1	1	3	2	1
officer_10	3	3	3	1	1	1	1
officer_100	1	1	1	1	1	2	3
officer_1000	2	3	1	3	3	3	3
officer_101	2	1	3	3	1	3	1
...	...	...	...	...	...	...	...
officer_995	2	2	2	3	2	2	3
officer_996	3	2	3	3	1	2	1
officer_997	3	2	1	3	1	3	2
officer_998	2	2	1	2	1	1	2
officer_999	2	1	3	3	2	1	3

Figure 8: Generated pivot table

Then using this pivot table, a sparse matrix has been created using scipy library. Then Brute Force algorithm was applied which is a simple version of KNN algorithm with the cosine matric as depicts in the Fig. 9. Finally, fitted the dataset which was converted to a matrix.



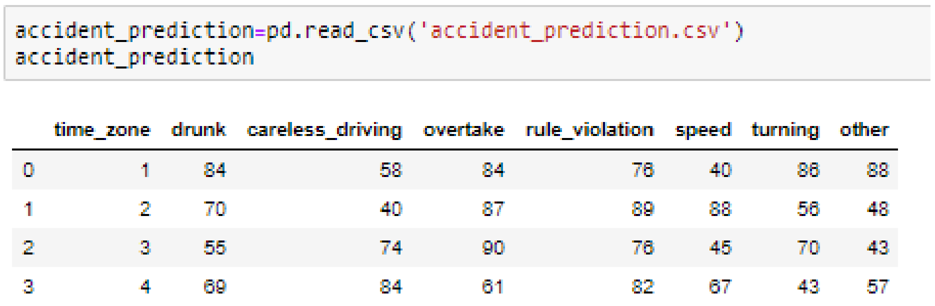
```
#convert the pivot table into an array matrix
from scipy.sparse import csr_matrix
officers_reasons_matrix = csr_matrix(officers_reasons_experience_rating.values)

#then import NearestNeighbors (unsupervised machine learning)
from sklearn.neighbors import NearestNeighbors

model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model_knn.fit(officers_reasons_matrix)
#default number of n_neighbors=5
#default p (euclidian distance = 2)
```

Figure 9: Implementing the Nearest Neighbors

The next step is the most important step of the whole model implementation process, which will implement the recommender system. In this application There were 4 recommendation systems were implemented for each time zone classified as road traffic accident prediction component. In order to perform the recommendation First of all a random officer needed to be selected from each time zone.

Selecting the random officers for each time zone

When selecting the random officer, it was considered to choose an officer with a higher experience rate for the top 3 reasons causing for occurring accidents in each time zone. Therefore first, a filtering approach was conducted to find the top most reasons for occurring accidents in each time zone by accessing the results of severity and reason predictions as follows in Fig. 10. And then for each 4 time zones, four random officers will be selected as mentioned before.



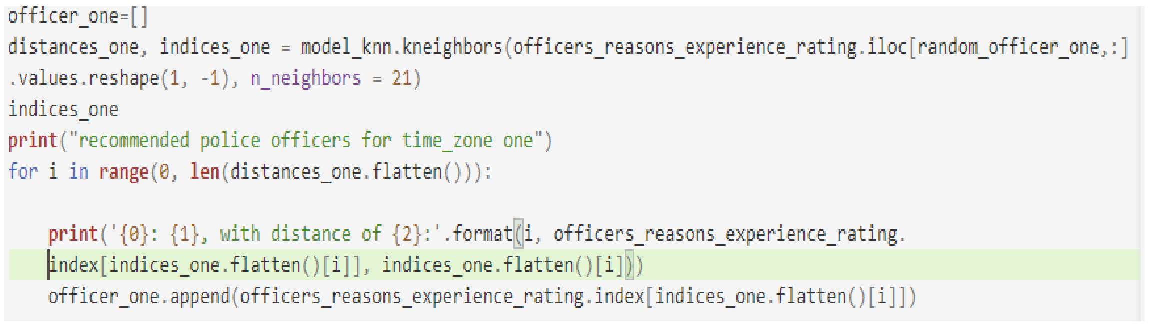
```
accident_prediction=pd.read_csv('accident_prediction.csv')
accident_prediction
```

	time_zone	drunk	careless_driving	overtake	rule_violation	speed	turning	other
0	1	84	58	84	76	40	86	88
1	2	70	40	87	89	88	56	48
2	3	55	74	90	76	45	70	43
3	4	69	84	61	82	67	43	57

Figure 10: Reading accident prediction results

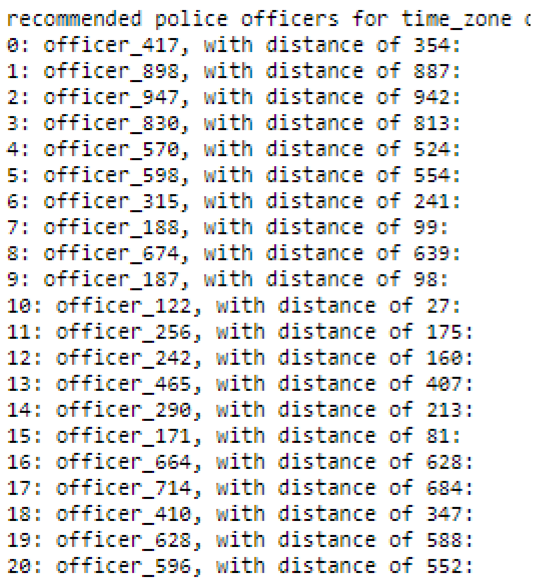
Developing the recommendation system

Based on the selected random officer, then the model will select similar officers for the given random officer. The number of officers that we want to recommend can be adjusted by the number of neighbors parameter of the model. According to the Code snippets depicts in Fig. 11, the recommender model will output the indices of the similar officers with the distance between them as in Fig. 12.



```
officer_one=[]
distances_one, indices_one = model_knn.kneighbors(officers_reasons_experience_rating.iloc[random_officer_one,:]
.values.reshape(1, -1), n_neighbors = 21)
indices_one
print("recommended police officers for time_zone one")
for i in range(0, len(distances_one.flatten())):

    print('{0}: {1}, with distance of {2}:'.format(i, officers_reasons_experience_rating.
    index[indices_one.flatten()[i]], indices_one.flatten()[i]))
    officer_one.append(officers_reasons_experience_rating.index[indices_one.flatten()[i]])
```

Figure 11: Recommender model implementation



```
recommended police officers for time_zone c
0: officer_417, with distance of 354:
1: officer_898, with distance of 887:
2: officer_947, with distance of 942:
3: officer_830, with distance of 813:
4: officer_570, with distance of 524:
5: officer_598, with distance of 554:
6: officer_315, with distance of 241:
7: officer_188, with distance of 99:
8: officer_674, with distance of 639:
9: officer_187, with distance of 98:
10: officer_122, with distance of 27:
11: officer_256, with distance of 175:
12: officer_242, with distance of 160:
13: officer_465, with distance of 407:
14: officer_290, with distance of 213:
15: officer_171, with distance of 81:
16: officer_664, with distance of 628:
17: officer_714, with distance of 684:
18: officer_410, with distance of 347:
19: officer_628, with distance of 588:
20: officer_596, with distance of 552:
```

Figure 11: recommended officers with distance values

The same methodology will be applied to recommend officers for the other time zones as well. And then the recommended officer details will be stores as a json file and send it to the database utilizing Flask.

Model implementation of police officer Scheduling system

Police officers scheduling was considered as an optimization problem and solved it by first implementing a mathematical model, then Used Gurobi optimizer, which is a commercial optimization solver considering the results of recommendation system.

Designing the mathematical Model

We have considered an area with *M* Check Points. The number of Police Officers required for Check Point *M* is denoted as $H_{m.}$ The number of Police Officers available in the area is denoted by *N*. For each Check Point m, we have allocated the Police Officers. The number of Police Officers allocated for each Check Point, must satisfy the required number of Police Officers for each Check Point. Maximum number of shifts per one officer within a week is denote by S. The duration of the generated Schedule is a week which is denote by D

We define z_n^{md} to be a binary variable for allocating Police Officer *n* for Check Point m within the day d, such that, if Police Officer *n* is allocated for Check Point *m* within the day *d*, then $z_n^{md} = 1$, otherwise it is 0.Table V provides a description of the key notations used for Police Officer allocation problem.

Table VI: Summary of Key Notations

N	No. of police officers, indexed by *n*=1,…,N (recommended for each time zone)
M	No. of checkpoints indexed by *m*=1,…M

z_n^{md}	Binary decision variable on selecting police officer *n* for checkpoint *m* in day *d*
S	Maximum number of shifts per one officer within one-week duration.
Hm	Required number of police officers for checkpoint *m*
D	Duration of the generating schedule, ranged from Monday to Sunday

The police officer allocation problem has solved using following mathematical model.

$$min \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{d=1}^{D} z_n^{md} \quad (1)$$

$$\sum_{m=1}^{M} \sum_{d=1}^{D} z_n^{md} \leq S, \forall n \quad (2)$$

$$\sum_{n=1}^{N} z_n^{md} \geq H_m, \forall m \quad (3)$$

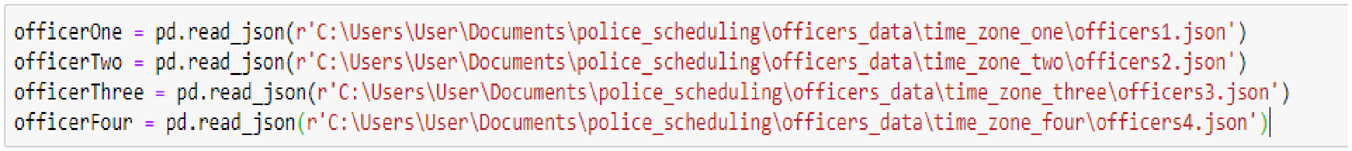
$$\sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{d=1}^{D} z_n^{md} \leq N * S \quad (4)$$

$$\sum_{m=1}^{M} z_n^{md} \leq 1 \quad (5)$$

The objective function (1) of the model ensure to minimize the number of shifts allocating for each officer within the schedule duration. Constraint (2) guarantees a police officer cannot allocated more than the maximum number of shifts a police officer can have within a week. Constraint (3) make sure the number of Police officers allocated for each checkpoint, satisfied the required number of police officers within each checkpoint. Constraint (4) guarantees the total number of Police officers allocated for all checkpoints within the schedule duration, should not exceeds the number of total number of possibilities of all the officers available. Constraint (5) ensures in one particular day, one Police officer can be allocated for only one or no checkpoint.

Implementing the mathematical model

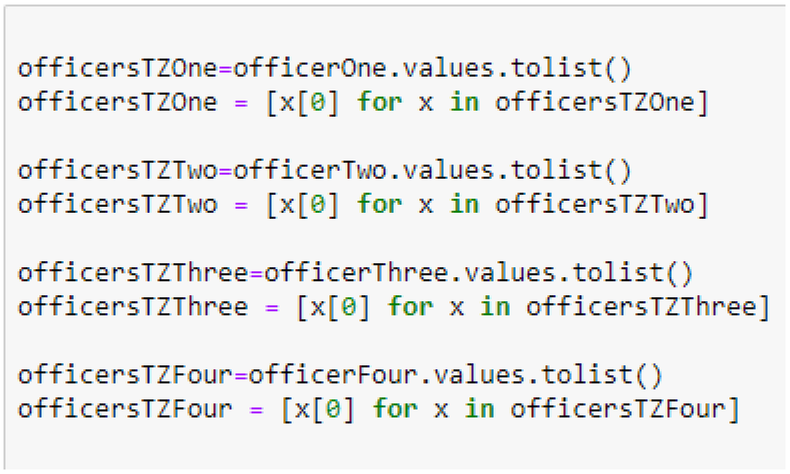
- ❖ Collecting the recommended officer details for each time zones.
  - o As depicts in Fig. 13. The recommended officer data were collected an converted them into data frames



```
officerOne = pd.read_json(r'C:\Users\User\Documents\police_scheduling\officers_data\time_zone_one\officers1.json')
officerTwo = pd.read_json(r'C:\Users\User\Documents\police_scheduling\officers_data\time_zone_two\officers2.json')
officerThree = pd.read_json(r'C:\Users\User\Documents\police_scheduling\officers_data\time_zone_three\officers3.json')
officerFour = pd.read_json(r'C:\Users\User\Documents\police_scheduling\officers_data\time_zone_four\officers4.json')
```

Figure 13: Collecting the recommended officer data sets

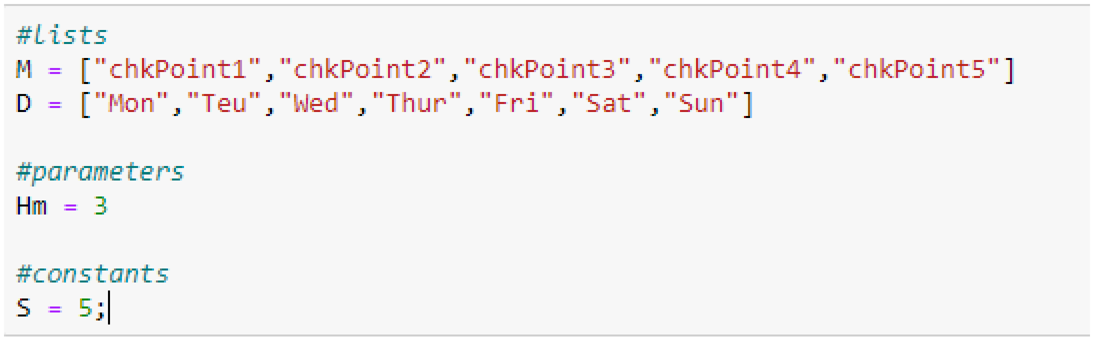
- ❖ Defining data lists.
  - o The created data frames are then converted into data lists as follows in Fig. 14.



```
officersTZOne=officerOne.values.tolist()
officersTZOne = [x[0] for x in officersTZOne]

officersTZTwo=officerTwo.values.tolist()
officersTZTwo = [x[0] for x in officersTZTwo]

officersTZThree=officerThree.values.tolist()
officersTZThree = [x[0] for x in officersTZThree]

officersTZFour=officerFour.values.tolist()
officersTZFour = [x[0] for x in officersTZFour]
```

Figure 14: generating the lists of recommended officers

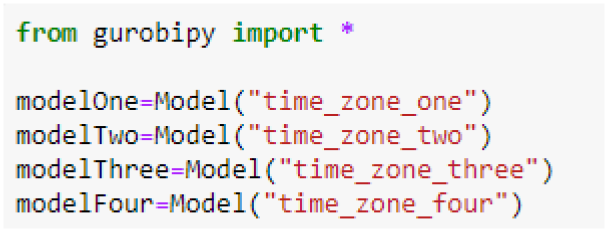
- Defining other sets, parameters and constants as mentioned in figure Fig. 15.



```
#lists
M = ["chkPoint1","chkPoint2","chkPoint3","chkPoint4","chkPoint5"]
D = ["Mon","Teu","Wed","Thur","Fri","Sat","Sun"]

#parameters
Hm = 3

#constants
S = 5;
```

Figure 15: Defining required parameters, lists and constants

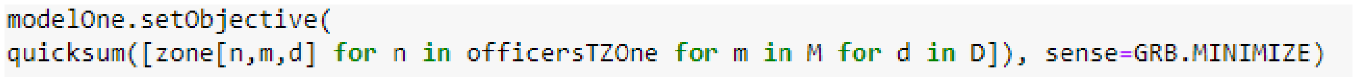
- Import gurobipy, the python wrapper to Gurobi solver and create new models for each tome zones as explained in Fig. 16.



```
from gurobipy import *

modelOne=Model("time_zone_one")
modelTwo=Model("time_zone_two")
modelThree=Model("time_zone_three")
modelFour=Model("time_zone_four")
```

Figure 16: Model generation for each time zone.

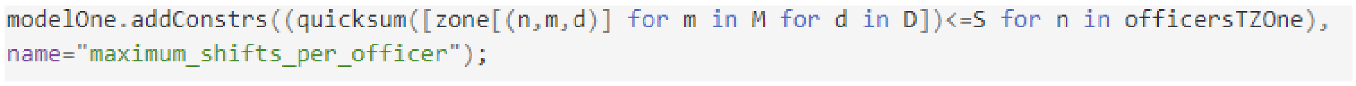
- From Now on the explanation will be based on the implementation of time zone one schedule. Same as this, the other three model will be implemented.
- Objective function implementation: Fig. 17.



```
modelOne.setObjective(
quicksum([zone[n,m,d] for n in officersTZOne for m in M for d in D]), sense=GRB.MINIMIZE)
```

Figure 17: Objective function implementation

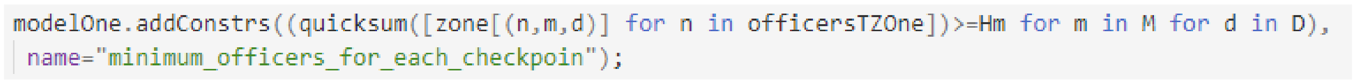
- Constraint 1 implementation depicts in Fig. 18.



```
modelOne.addConstrs((quicksum([zone[(n,m,d)] for m in M for d in D])<=S for n in officersTZOne),
name="maximum_shifts_per_officer");
```

Figure 18: Constraint 1 implementation

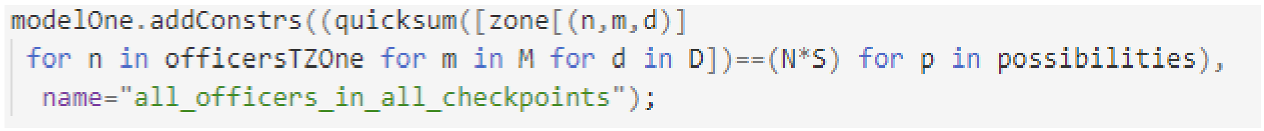
- Constraint 2 implementation explain in Fig. 19.



```
modelOne.addConstrs((quicksum([zone[(n,m,d)] for n in officersTZOne])>=Hm for m in M for d in D),
 name="minimum_officers_for_each_checkpoin");
```

Figure 19: Constraint 2 implementation

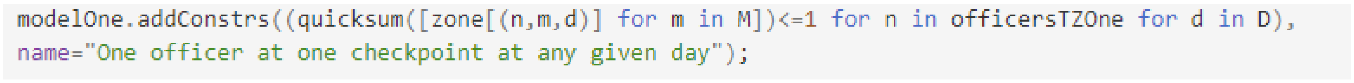
- Constraint 3 implementation described in Fig. 20.



```
modelOne.addConstrs((quicksum([zone[(n,m,d)]
 for n in officersTZOne for m in M for d in D])==(N*S) for p in possibilities),
  name="all_officers_in_all_checkpoints");
```

Figure 20: Constraint 3 implementation

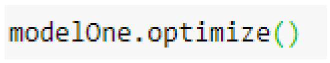
- Constraint 4 implementation explained in Fig. 21.



```
modelOne.addConstrs((quicksum([zone[(n,m,d)] for m in M])<=1 for n in officersTZOne for d in D),
name="One officer at one checkpoint at any given day");
```

Figure 21: Constraint 4 implementation

- Then the generated model will perform the optimization as bellow in Fig. 22.



```
modelOne.optimize()
```

Figure 21: Perform optimization

Functional implementation

Architectural style

The SmartCop web application has been implemented according to the MVC architectural style maintaining its quality. According to the architecture, the software application is divided into three sections such as the Model, View, and Controller. Model is the component where the data-related work is handled. Also, the View is the section where the user accesses the application or generally it is the frontend. Nevertheless, the Controller handles the business logic of the application acting as an interface between the Model and the View.

Sign-Up process

Sign Up form is implemented using ReactJS and it requires fields,

- First Name
- Last Name
- NIC
- Contact number
- Police ID
- Region
- Police station
- Email
- Password

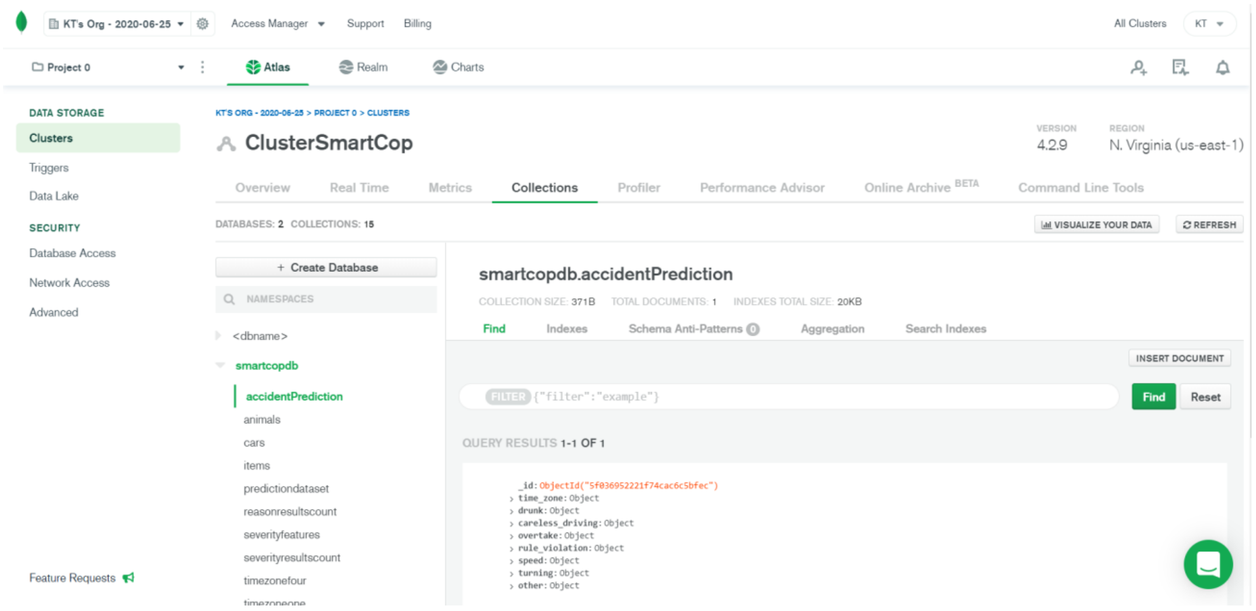
After the relevant user enters the required data, form will be submitted after the validation process. If there are validation issues, the user would be advised to insert valid data. If the inserted data is correct, user profile would be automatically created. Therefore, the authorized police officer can Sign-In to the SmartCop web application.

Sign-In process

After creating the user profile in the Sign-Up process, user can Sign-In to the SmartCop web application. In order to proceed, user needs to enter the registered email address as the username and the password. When user clicks on the submit button, the system validates them considering the data in database. If there is a credential miss match, the system would prompt an error message otherwise the user can successfully log into the system and access the SmartCop main dashboard.

Database connection

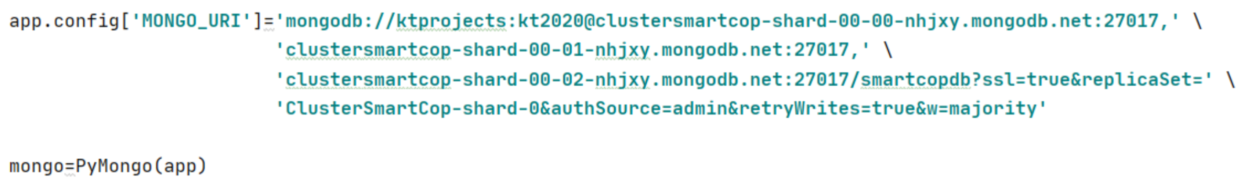
MongoDB database has been used as the datastore of the SmartCop web application. Therefore, all data relevant to the application is stored inside the MongoDB Atlas Cloud. After creating a cluster inside the MongoDB Atlas Cloud as shown in Fig. 22, the connection has been established in the application itself as shown in the Fig. 23.



```
{"filter":"example"}
```

```
  _id: ObjectId("5f036952221f74cac6c5bfec")
> time_zone: Object
> drunk: Object
> careless_driving: Object
> overtake: Object
> rule_violation: Object
> speed: Object
> turning: Object
> other: Object
```

Figure 22: MongoDB Atlas Cloud SmartCop cluster



```
app.config['MONGO_URI']='mongodb://ktprojects:kt2020@clustersmartcop-shard-00-00-nhjxy.mongodb.net:27017,' \
                        'clustersmartcop-shard-00-01-nhjxy.mongodb.net:27017,' \
                        'clustersmartcop-shard-00-02-nhjxy.mongodb.net:27017/smartcopdb?ssl=true&replicaSet=' \
                        'ClusterSmartCop-shard-0&authSource=admin&retryWrites=true&w=majority'

mongo=PyMongo(app)
```

Figure 23: Database connection

3.RESULTS & DISCUSSION

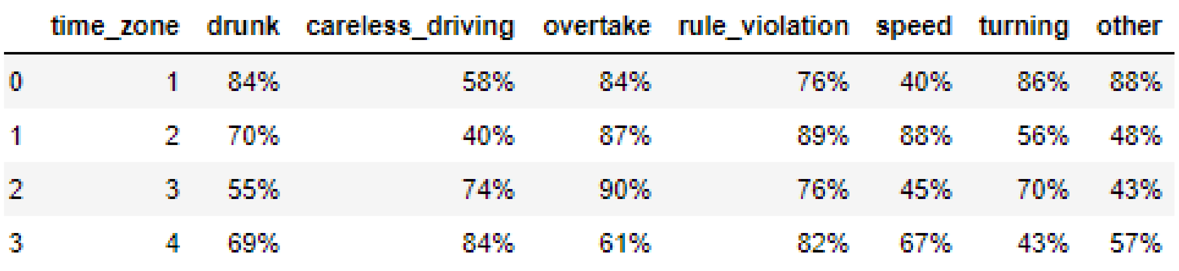
3.1 Results

This section elaborates the results and observations of traffic police officers recommendation and scheduling component in the SmartCop project. The results going to be discussed under this section, have obtained from the recommendation and scheduling process delineate in the methodology section. Here, the results are going to be discussed under two main categories respectively, traffic police officers recommendation results and Traffic police officers scheduling results.

3.1.1 Traffic police officers recommendation results

As mentioned in the Methodology section, the number of accidents happened due to each reason, at each time zone has been taken into consideration to do the recommendation. To check the performances of our approach, we have assumed three levels for experience of a police officer: level 1, level 2 and level 3, where level 3 is the highest experience level. Each police officer will be assigned an experience level for each accident causing reason (careless driving, drunk driving, high speed, overtake, rules violation, turning, and any other factors), based on their experiences on handling accidents caused by these reasons.

According to the reason and severity prediction results gain from the first component of the SmartCop web application as depicts in Fig. 24, each time zone has top 3 reasons which caused for occurring accidents.



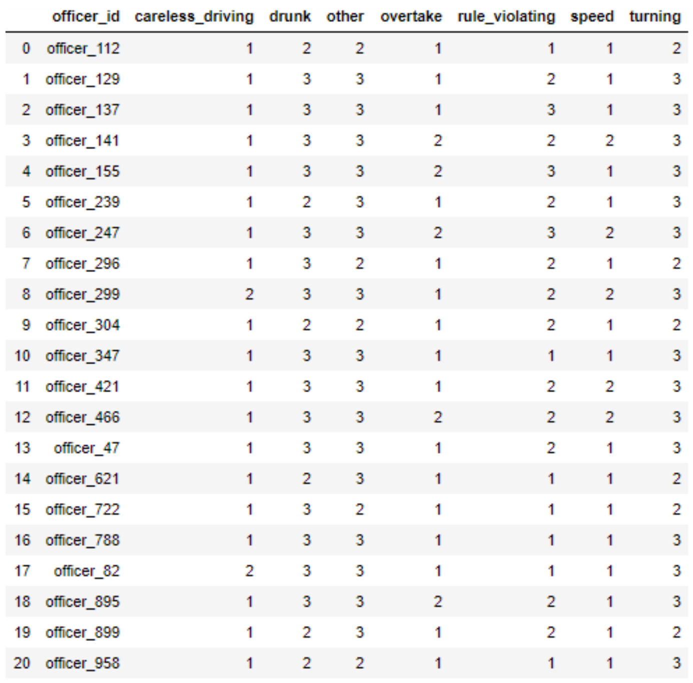
	time_zone	drunk	careless_driving	overtake	rule_violation	speed	turning	other
0	1	84%	58%	84%	76%	40%	86%	88%
1	2	70%	40%	87%	89%	88%	56%	48%
2	3	55%	74%	90%	76%	45%	70%	43%
3	4	69%	84%	61%	82%	67%	43%	57%

Figure 24: Road accident reason and severity prediction results

As illustrates in the Fig. 24,

- Top three reasons for occurring accidents in time zone one is
  - Other reasons (88%)
  - Turning (86%)
  - Drunk (84%)
- Top three reasons for occurring accidents in time zone two is
  - Rule violation (89%)
  - Speed (88%)
  - Overtake (87%)
- Top three reasons for occurring accidents in time zone three is
  - Overtake (90%)
  - Rule violation (76%)
  - Careless driving (74%)
- Top three reasons for occurring accidents in time zone four is
  - Careless driving (84%)
  - Rule violating (82%)
  - Drunk (69%)

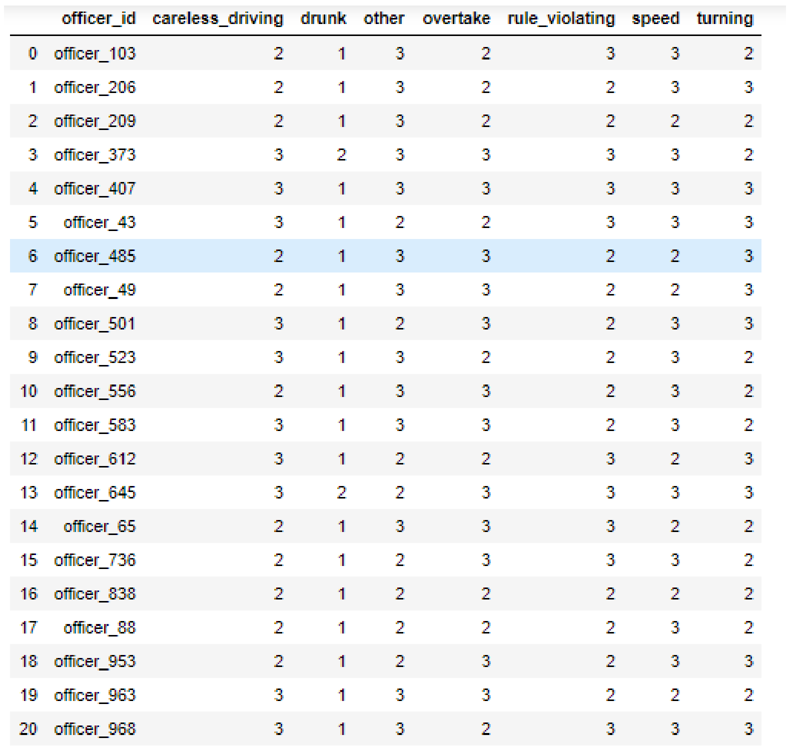
As discussed in methodology section, when selecting the random officer in order to do the recommendation, it was considered to choose an officer with a higher experience rate (level 3) for the top 3 reasons causing for occurring accidents in each time zone. Therefore, the recommended results also should have officers with good experience for the top 3 reasons causing occur accidents in each time zone. Fig. 25 illustrates the recommended officer results for time zone one. According to that we can see for turning, drunk and Other reasons which are the top 3 reasons caused for occurring accidents in time zone one, the recommended officers are only having the experience of level 3 and 2. There is no officer recommended with experience level 1 for the mentioned reasons.



	officer_id	careless_driving	drunk	other	overtake	rule_violating	speed	turning
0	officer_112	1	2	2	1	1	1	2
1	officer_129	1	3	3	1	2	1	3
2	officer_137	1	3	3	1	3	1	3
3	officer_141	1	3	3	2	2	2	3
4	officer_155	1	3	3	2	3	1	3
5	officer_239	1	2	3	1	2	1	3
6	officer_247	1	3	3	2	3	2	3
7	officer_296	1	3	2	1	2	1	2
8	officer_299	2	3	3	1	2	2	3
9	officer_304	1	2	2	1	2	1	2
10	officer_347	1	3	3	1	1	1	3
11	officer_421	1	3	3	1	2	2	3
12	officer_466	1	3	3	2	2	2	3
13	officer_47	1	3	3	1	2	1	3
14	officer_621	1	2	3	1	1	1	2
15	officer_722	1	3	2	1	1	1	2
16	officer_788	1	3	3	1	1	1	3
17	officer_82	2	3	3	1	1	1	3
18	officer_895	1	3	3	2	2	1	3
19	officer_899	1	2	3	1	2	1	2
20	officer_958	1	2	2	1	1	1	3

Figure 25: Recommended officer results for time zone one.

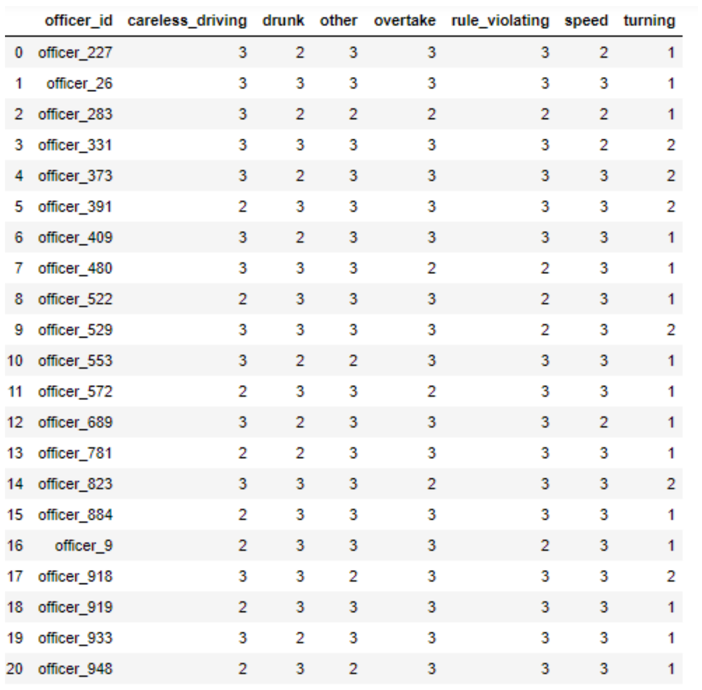
Then, Fig. 26 illustrates the recommended officer results for time zone two. According to that we can see for Rule Violation, speed and overtake which are the top 3 reasons caused for occurring accidents in time zone two, the recommended officers are only having the experience of level 3 and 2. There is no officer recommended with experience level 1 for the mentioned reasons.



	officer_id	careless_driving	drunk	other	overtake	rule_violating	speed	turning
0	officer_103	2	1	3	2	3	3	2
1	officer_206	2	1	3	2	2	3	3
2	officer_209	2	1	3	2	2	2	2
3	officer_373	3	2	3	3	3	3	2
4	officer_407	3	1	3	3	3	3	3
5	officer_43	3	1	2	2	3	3	3
6	officer_485	2	1	3	3	2	2	3
7	officer_49	2	1	3	3	2	2	3
8	officer_501	3	1	2	3	2	3	3
9	officer_523	3	1	3	2	2	3	2
10	officer_556	2	1	3	3	2	3	2
11	officer_583	3	1	3	3	2	3	2
12	officer_612	3	1	2	2	3	2	3
13	officer_645	3	2	2	3	3	3	3
14	officer_65	2	1	3	3	3	2	2
15	officer_736	2	1	2	3	3	3	2
16	officer_838	2	1	2	2	2	2	2
17	officer_88	2	1	2	2	2	3	2
18	officer_953	2	1	2	3	2	3	3
19	officer_963	3	1	3	3	2	2	2
20	officer_968	3	1	3	2	3	3	3

Figure 26: Recommended officer results for time zone two.

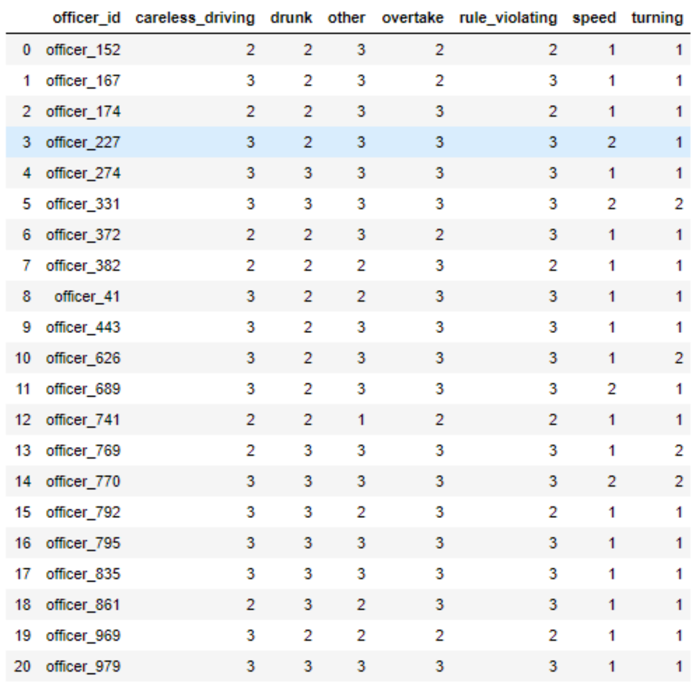
Then, Fig. 27 illustrates the recommended officer results for time zone three. According to that we can see for Rule Violation, Careless driving and overtake which are the top 3 reasons caused for occurring accidents in time zone three, the recommended officers are only having the experience of level 3 and 2. There is no officer recommended with experience level 1 for the mentioned reasons.



	officer_id	careless_driving	drunk	other	overtake	rule_violating	speed	turning
0	officer_227	3	2	3	3	3	2	1
1	officer_26	3	3	3	3	3	3	1
2	officer_283	3	2	2	2	2	2	1
3	officer_331	3	3	3	3	3	2	2
4	officer_373	3	2	3	3	3	3	2
5	officer_391	2	3	3	3	3	3	2
6	officer_409	3	2	3	3	3	3	1
7	officer_480	3	3	3	2	2	3	1
8	officer_522	2	3	3	3	2	3	1
9	officer_529	3	3	3	3	2	3	2
10	officer_553	3	2	2	3	3	3	1
11	officer_572	2	3	3	2	3	3	1
12	officer_689	3	2	3	3	3	2	1
13	officer_781	2	2	3	3	3	3	1
14	officer_823	3	3	3	2	3	3	2
15	officer_884	2	3	3	3	3	3	1
16	officer_9	2	3	3	3	2	3	1
17	officer_918	3	3	2	3	3	3	2
18	officer_919	2	3	3	3	3	3	1
19	officer_933	3	2	3	3	3	3	1
20	officer_948	2	3	2	3	3	3	1

Figure 27: Recommended officer results for time zone three.

Then, Fig. 28 illustrates the recommended officer results for time zone four. According to that we can see for Rule Violation, Careless driving and drunk which are the top 3 reasons caused for occurring accidents in time zone four, the recommended officers are only having the experience of level 3 and 2. There is no officer recommended with experience level 1 for the mentioned reasons.



	officer_id	careless_driving	drunk	other	overtake	rule_violating	speed	turning
0	officer_152	2	2	3	2	2	1	1
1	officer_167	3	2	3	2	3	1	1
2	officer_174	2	2	3	3	2	1	1
3	officer_227	3	2	3	3	3	2	1
4	officer_274	3	3	3	3	3	1	1
5	officer_331	3	3	3	3	3	2	2
6	officer_372	2	2	3	2	3	1	1
7	officer_382	2	2	2	3	2	1	1
8	officer_41	3	2	2	3	3	1	1
9	officer_443	3	2	3	3	3	1	1
10	officer_626	3	2	3	3	3	1	2
11	officer_689	3	2	3	3	3	2	1
12	officer_741	2	2	1	2	2	1	1
13	officer_769	2	3	3	3	3	1	2
14	officer_770	3	3	3	3	3	2	2
15	officer_792	3	3	2	3	2	1	1
16	officer_795	3	3	3	3	3	1	1
17	officer_835	3	3	3	3	3	1	1
18	officer_861	2	3	2	3	3	1	1
19	officer_969	3	2	2	2	2	1	1
20	officer_979	3	3	3	3	3	1	1

Figure 29: Recommended officer results for time zone four.

Afterwards, we have conducted the same experiment by selecting the random officer with experience level 2.

In the first experiment (for the level 3 random officer) average results for all time zones showed for 98.33% of the experiments runs, the system allocated police officers with experience level 3 and 2, for top 3 types of accidents within the relevant time zones. For 1.67% of the experiments runs, the combination of police officers recommended by the system for all time zones, included police officers with level 1 experiences and that can be clearly seen in the Fig. 30.

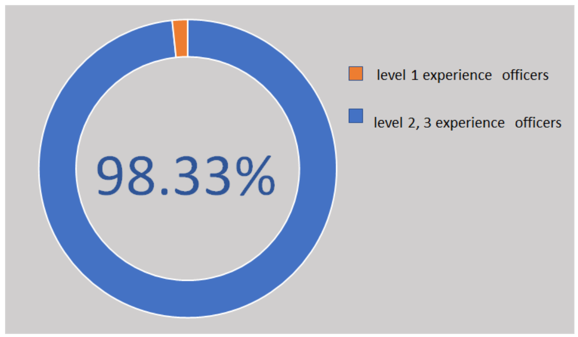


Figure 30: Overall experiment results for selecting level 3 random officer

In the second experiment (for the level 2 random officer) average results for all time zones showed for 94.21% of the experiments runs, the system allocated police officers with experience level 3 and 2, for top 3 types of accidents within the relevant time zones. For 5.79% of the experiments runs, the combination of police officers recommended by the system for all time zones, included police officers with level 1 experiences and that can be clearly seen in the Fig. 31.

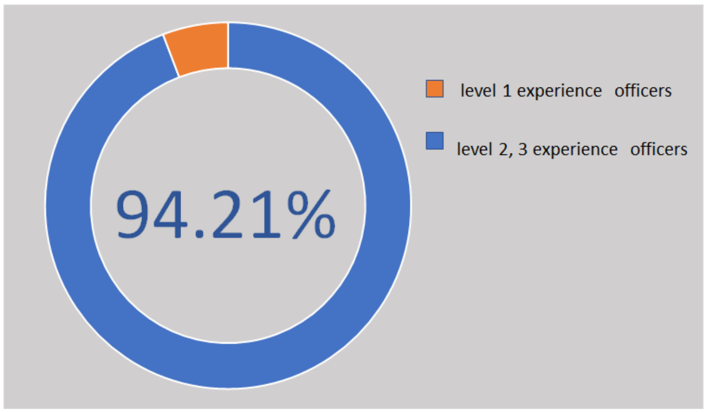


Figure 31: Overall experiment results for selecting random officer with level 2 experience

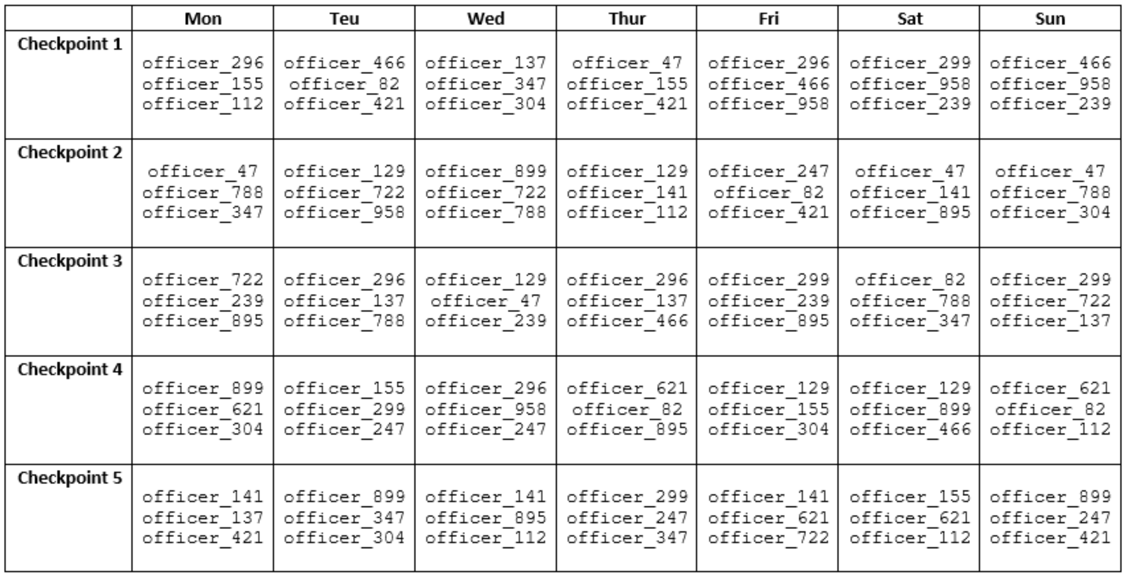
3.1.1 Traffic police officers Scheduling results

As mentioned in the methodology section, the recommended officers then taken by the scheduling component to schedule them optimally within the duration of seven days. When scheduling the implemented mathematical model ensures to schedule officers by fulfilling the objective and constraints given.

- Objective: minimize the number of shifts allocating for each officer within the schedule duration.
- Constraint 1: an officer cannot be allocated more than the maximum number of shifts a police officer can have within a week.
- Constraint 2: number of Police officers allocated for each checkpoint, satisfied the required number of police officers within each checkpoint
- Constraint 3: total number of Police officers allocated for all checkpoints within the schedule duration, should not exceeds the number of total number of possibilities of all the officers available

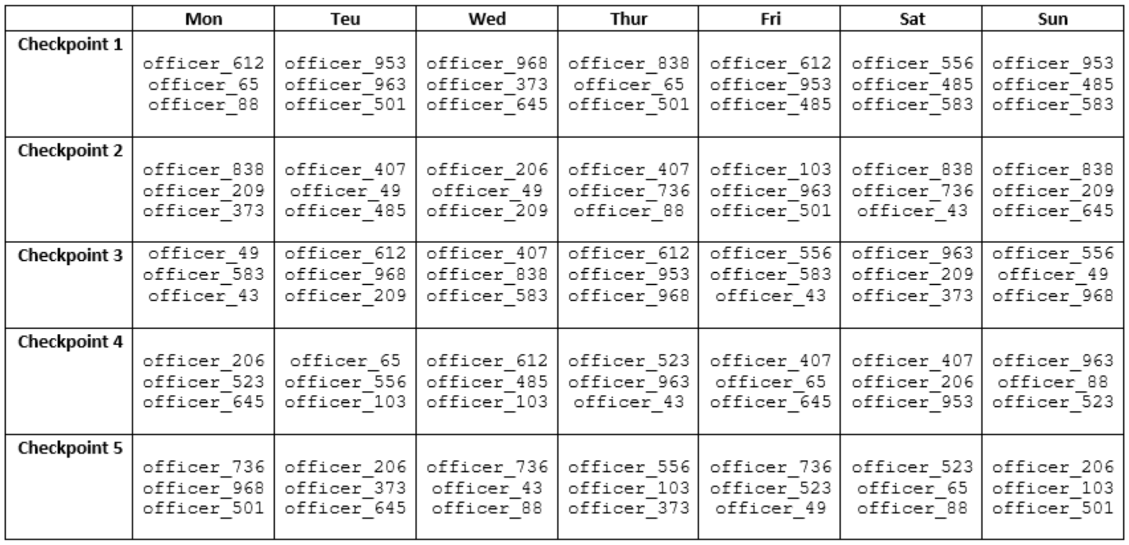
- Constraint 4: ensures in one particular day, one Police officer can be allocated for only one or no checkpoint.

By following the given objective function and all the constraints, the police officers schedule for the recommended officer for each time zones, respectively for time zone one to four can be seen in Fig. 26 to Fig 29. As depicts in the following figures, it can be clearly seen that the officers in each schedule are same as the recommended officers in each time zone. And also, the schedule has been done by considering all the rules mentioned in the constrains.



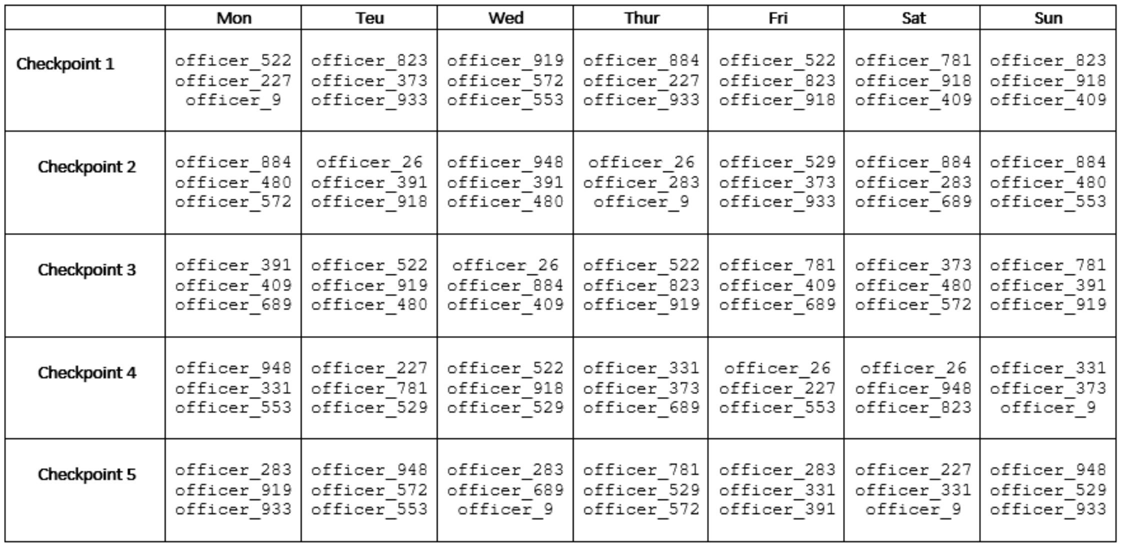
	Mon	Teu	Wed	Thur	Fri	Sat	Sun
Checkpoint 1	officer_296 officer_155 officer_112	officer_466 officer_82 officer_421	officer_137 officer_347 officer_304	officer_47 officer_155 officer_421	officer_296 officer_466 officer_958	officer_299 officer_958 officer_239	officer_466 officer_958 officer_239
Checkpoint 2	officer_47 officer_788 officer_347	officer_129 officer_722 officer_958	officer_899 officer_722 officer_788	officer_129 officer_141 officer_112	officer_247 officer_82 officer_421	officer_47 officer_141 officer_895	officer_47 officer_788 officer_304
Checkpoint 3	officer_722 officer_239 officer_895	officer_296 officer_137 officer_788	officer_129 officer_47 officer_239	officer_296 officer_137 officer_466	officer_299 officer_239 officer_895	officer_82 officer_788 officer_347	officer_299 officer_722 officer_137
Checkpoint 4	officer_899 officer_621 officer_304	officer_155 officer_299 officer_247	officer_296 officer_958 officer_247	officer_621 officer_82 officer_895	officer_129 officer_155 officer_304	officer_129 officer_899 officer_466	officer_621 officer_82 officer_112
Checkpoint 5	officer_141 officer_137 officer_421	officer_899 officer_347 officer_304	officer_141 officer_895 officer_112	officer_299 officer_247 officer_347	officer_141 officer_621 officer_722	officer_155 officer_621 officer_112	officer_899 officer_247 officer_421

Figure 26: Officers schedule result for time zone one



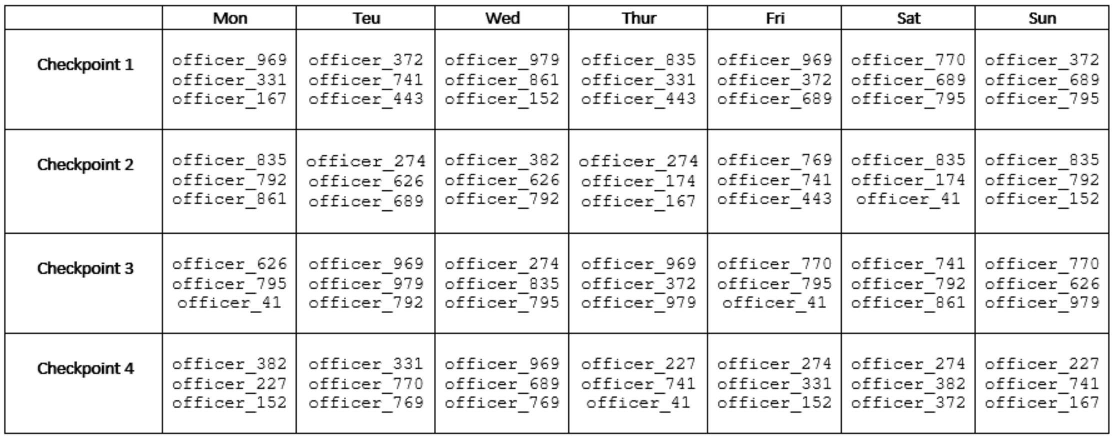
	Mon	Teu	Wed	Thur	Fri	Sat	Sun
Checkpoint 1	officer_612 officer_65 officer_88	officer_953 officer_963 officer_501	officer_968 officer_373 officer_645	officer_838 officer_65 officer_501	officer_612 officer_953 officer_485	officer_556 officer_485 officer_583	officer_953 officer_485 officer_583
Checkpoint 2	officer_838 officer_209 officer_373	officer_407 officer_49 officer_485	officer_206 officer_49 officer_209	officer_407 officer_736 officer_88	officer_103 officer_963 officer_501	officer_838 officer_736 officer_43	officer_838 officer_209 officer_645
Checkpoint 3	officer_49 officer_583 officer_43	officer_612 officer_968 officer_209	officer_407 officer_838 officer_583	officer_612 officer_953 officer_968	officer_556 officer_583 officer_43	officer_963 officer_209 officer_373	officer_556 officer_49 officer_968
Checkpoint 4	officer_206 officer_523 officer_645	officer_65 officer_556 officer_103	officer_612 officer_485 officer_103	officer_523 officer_963 officer_43	officer_407 officer_65 officer_645	officer_407 officer_206 officer_953	officer_963 officer_88 officer_523
Checkpoint 5	officer_736 officer_968 officer_501	officer_206 officer_373 officer_645	officer_736 officer_43 officer_88	officer_556 officer_103 officer_373	officer_736 officer_523 officer_49	officer_523 officer_65 officer_88	officer_206 officer_103 officer_501

Figure 27: Officers schedule result for time zone two



	Mon	Teu	Wed	Thur	Fri	Sat	Sun
Checkpoint 1	officer_522 officer_227 officer_9	officer_823 officer_373 officer_933	officer_919 officer_572 officer_553	officer_884 officer_227 officer_933	officer_522 officer_823 officer_918	officer_781 officer_918 officer_409	officer_823 officer_918 officer_409
Checkpoint 2	officer_884 officer_480 officer_572	officer_26 officer_391 officer_918	officer_948 officer_391 officer_480	officer_26 officer_283 officer_9	officer_529 officer_373 officer_933	officer_884 officer_283 officer_689	officer_884 officer_480 officer_553
Checkpoint 3	officer_391 officer_409 officer_689	officer_522 officer_919 officer_480	officer_26 officer_884 officer_409	officer_522 officer_823 officer_919	officer_781 officer_409 officer_689	officer_373 officer_480 officer_572	officer_781 officer_391 officer_919
Checkpoint 4	officer_948 officer_331 officer_553	officer_227 officer_781 officer_529	officer_522 officer_918 officer_529	officer_331 officer_373 officer_689	officer_26 officer_227 officer_553	officer_26 officer_948 officer_823	officer_331 officer_373 officer_9
Checkpoint 5	officer_283 officer_919 officer_933	officer_948 officer_572 officer_553	officer_283 officer_689 officer_9	officer_781 officer_529 officer_572	officer_283 officer_331 officer_391	officer_227 officer_331 officer_9	officer_948 officer_529 officer_933

Figure 28: Officers schedule result for time zone three



	Mon	Teu	Wed	Thur	Fri	Sat	Sun
Checkpoint 1	officer_969 officer_331 officer_167	officer_372 officer_741 officer_443	officer_979 officer_861 officer_152	officer_835 officer_331 officer_443	officer_969 officer_372 officer_689	officer_770 officer_689 officer_795	officer_372 officer_689 officer_795
Checkpoint 2	officer_835 officer_792 officer_861	officer_274 officer_626 officer_689	officer_382 officer_626 officer_792	officer_274 officer_174 officer_167	officer_769 officer_741 officer_443	officer_835 officer_174 officer_41	officer_835 officer_792 officer_152
Checkpoint 3	officer_626 officer_795 officer_41	officer_969 officer_979 officer_792	officer_274 officer_835 officer_795	officer_969 officer_372 officer_979	officer_770 officer_795 officer_41	officer_741 officer_792 officer_861	officer_770 officer_626 officer_979
Checkpoint 4	officer_382 officer_227 officer_152	officer_331 officer_770 officer_769	officer_969 officer_689 officer_769	officer_227 officer_741 officer_41	officer_274 officer_331 officer_152	officer_274 officer_382 officer_372	officer_227 officer_741 officer_167

Figure 29: Officers schedule result for time zone four

3.2 Research Findings

The smart cup is an automated platform implemented to mitigate traffic accidents. As the second component police officers recommendation and scheduling component deliver a unique and innovative solution when compared with the existing researches and products as this component not only considering scheduling officers, instead it recommends the suitable officer for the suitable job and then optimally schedule the recommended officers in a way that minimizes the number of shifts allocating for each officer within a week which is indirectly dissolves in a cost-effective schedule with most appropriate officers.

moreover, the recommendation system implemented in SmartCop has been followed a collaborative filtering approach to recommend traffic police officers. According to the definition [1] collaborative filtering is a technique that can filter out items that a user might like on the basis similar uses. But by differentiating from the typical collaborative approach in smart the collaborative filtering approach was conducted by filtering out users by considering the similarity with the other uses. Therefore not like in typical collaborative recommendation system which recommended items or jobs to users, this system recommend users to jobs which is the opposite way of typical collaborative recommendation system.

In addition, it was found that it is important to train the model several times to improve the performance of the model and increase the accuracy of the results generating from the model. Furthermore, to implement the recommendation model in SmartCop it did not use only the KNN algorithm but also it was used cosine similarity to calculate the distance between similar users in order to get better and accurate results than the results of using only the algorithm.

3.3 Discussion

The finalized research project has implemented with the intention of mitigating the impact of road traffic accidents by including four main components respectively, predict road accidents, recommend and schedule police officers, enhance road accidents prevention awareness, and enhance road accidents response awareness. Further, the SmartCop has developed as a combination of two sub platforms as web-based application which consists the road accident prediction, and police officers' recommendation and scheduling components. And the other sub platform is a game-based learning mobile application which contains the road accidents prevention awareness game, and the road accidents response awareness game. Among them this component comprises the traffic police officers recommendation and scheduling system.

Accurate traffic police officer scheduling is a complex problem in locally as well as globally and Poor scheduling of officers does not make any sense of reducing the rate of traffic accidents. Even though several statistical applications and researches have conducted, there is still a necessity for the development of an improved system for scheduling and enforcement of traffic police officers. As the solution for the observed problem we have developed an automated platform called "SmartCop" which comprises a module to properly schedule traffic police officers with a recommendation system.

Hence, the traffic police officers recommendation and scheduling component comprises two main sub components as, recommending traffic police officers based on their experience analysis and then recommended officers were scheduled to available checkpoints by considering different time zones predicted from the accident prediction component. Also, the outcomes are analyzed and delivered to the end user via the SmartCop web application. Apart from the mentioned sub components the other major functionalities of this application are,

- ➢ View recommended officer details for each time zone

- View scheduled officer details for each time zone.
- View road traffic accident summary details of severity and reason prediction taken from accident prediction component
- View officer details
- Send notifications for the scheduled officers

Furthermore, Python ReactJS with HTML, CSS, and bootstrap have used as main technologies to implement the backend and front end of the web application. PyCharm and VSCode was used as IDE while MongoDB was utilized as the database when implementing the application. In order to implement the recommendation system a collaborative filtering approach has been used with KNN algorithm and the Cosine similarity technique. Then a mathematical model was proposed based on optimization and implemented utilizing the Gurobi Optimizer which is a commercial optimization solver. The major challenges encountered with when developing the application are finding a methodology to develop the recommendation system that recommend users to jobs which is the opposite way of a typical recommendation system and the other challenge was to design the mathematical which needed to schedule officers optimally.

However, a completed component for recommending ad scheduling traffic police officers has been developed successfully along with a web application to deliver the overall content. As this project is in the initial phase the scheduling component was designed to a single police station. Nevertheless, It can be expanded for multiple police stations as per the requirements of the clients in the future.

4. CONCLUSION

With the intention of mitigation, the impact of road traffic accidents an automated platform has proposed and developed which comprises two sub platform as a web application and a mobile application. Hence by concentrating on the second component of the SmartCop research project, this report elaborates the development of traffic police officers recommendation and scheduling system.

Proper traffic police officers scheduling platform is a vast requirement of not only Sri Lanka, but all over the world. Even though several statistical applications available, there is still a necessity for the development of an improved system for scheduling and enforcement of traffic police officers since it was not considered the suitability of officers to the job which they have scheduled in the existing researches as depicts in the literature review section. In order to fill the gap SmartCop was came Up with a proper solution which recommend the traffic police officers based on their experience analysis and schedule the recommended officers optimally by designing and implementing a mathematical model based on optimization.

The recommendation of officers was done for 4 separate time zones respectively, time zone 1 (12.00 am – 6.00 am), time zone 2 (6.00 am – 12.00 pm), time zone 3 (12.00 pm – 6.00 pm), and time zone 4 (6.00 pm – 12.00 pm) which was predicted and classified from the road traffic accident prediction component. The system will analyze the top 3 reasons for occurring accidents in each time zone, then recommend officers who have better experience on those reasons. In order to perform the recommendation KNN algorithm along with the Cosine similarity was used. Then the recommended officers will be scheduling to available checkpoints and time zones as mentioned before by considering the scheduling officers as an optimization problem.

In accordance to the commercialization aspect the scheduling component will be given as the free trial to the client and to get improved performance recommendation module packages as the premium version. Furthermore, more features can be added and expand the application according to the clients' requirements in the future. In conclusion the SmartCop police officer's recommendation and scheduling component ensures the required service for all police stations in order to minimize the impact of road traffic accidents.

.

REFERECES

[1] (2020, 23-April-2020). *Road traffic injuries*. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries#:~:text=Approximately%201.35%20million%20people%20die,of%20their%20gross%20domestic%20product.

[2] M. Másilková, "Health and social consequences of road traffic accidents," *Kontakt*, vol. 19, no. 1, pp. e43-e47, 2017/03/01/ 2017.

[3] R. Gorea, "Financial impact of road traffic accidents on the society," *International Journal of Ethics, Trauma & Victimology*, vol. 2, 07/29 2016.

[4] S.-h. Park, S.-m. Kim, and Y.-g. Ha, "Highway traffic accident prediction using VDS big data analysis," *The Journal of Supercomputing,* vol. 72, 01/20 2016.

[5] V. M. Ramachandiran, P. N. K. Babu, and R. Manikandan, "Prediction of road accidents severity using various algorithms," *International Journal of Pure and Applied Mathematics,* vol. 119, pp. 16663-16669, 01/01 2018.

[6] K. R. Sumana and D. Phaneendra, *Smart Automated Modelling using Eclat Algorithm for Traffic Accident Prediction*. 2019.

[7] A. Ashtaiwi, "Intelligent Road Crashes Avoidance System," 2019.

[8] F. Labib, A. Rifat, M. Hossain, A. Das, and F. Nawrine, "Road Accident Analysis and Prediction of Accident Severity by Using Machine Learning in Bangladesh," in *7th International Conference on Smart Computing & Communications*, 2019, pp. 1-5.

[9] G. Kaur, E. H. J. t. I. C. o. C. Kaur, Communication, and N. Technologies, "Prediction of the cause of accident and accident prone location on roads using data mining techniques," pp. 1-7, 2017.

[10] E. Reveron and A. Cretu, "A Framework for Collision Prediction Using Historical Accident Information and Real-time Sensor Data: A Case Study for the City of Ottawa," in *2019 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, 2019, pp. 1-7.

[11] Q. Liyan and S. Chunfu, *Macro Prediction Model of Road Traffic Accident Based on Neural Network and Genetic Algorithm*. 2009, pp. 354-357.

[12] A. J. Graettinger, J. K. Lindly, and G. J. Mistry, "Display and analysis of crash data," (in English), Research Paper 2005.

[13] T. Sivakumar and D. Amarathung, "Development of Traffic Accident Prediction Models Using Traffic and Road Characteristics : A Case Study from Sri Lanka," 2015.

[14] R. G. Ramani, S. J. I. I. C. o. C. I. Shanthi, and C. Research, "Classifier prediction evaluation in modeling road traffic accident data," pp. 1-4, 2012.

[15] JetBrains. (1-April-2020). *PyCharm*. Available: https://www.jetbrains.com/pycharm/

[16] (1-April-2020). *Visual Studio Code*. Available: https://code.visualstudio.com/docs/editor/whyvscode

[17] (1-Jan-2020). *What is Python? Executive Summary*. Available: https://www.python.org/doc/essays/blurb/

[18] (23-March-2020). *Flask (web framework)*. Available: https://en.wikipedia.org/wiki/Flask_(web_framework)

[19] (1-March-2020). *MongoDB Atlas*. Available: https://docs.atlas.mongodb.com/

[20] "Beginners Guide to ReactJS," Accessed on: 23-April-2020Available: https://medium.com/zenofai/beginners-guide-to-reactjs-3ca07f56d526#:~:text=React%20or%20ReactJS%20is%20a%20JavaScript%20library.&text=React%20is%20an%20open%2Dsource,application%20(Model%20View%20Controller).

[21] (25-April-2020). *Anaconda Navigator*. Available: https://docs.anaconda.com/anaconda/navigator/#:~:text=Anaconda%20Navigator%20is%20a%20desktop,without%20using%20command%2Dline%20commands.&text=To%20get%20Navigator%2C%20get%20the%20Navigator%20Cheat%20Sheet%20and%20install%20Anaconda.

[22] (2020, 12-July-2020). *Commercialization*. Available: https://www.investopedia.com/terms/c/commercialization.asp#:~:text=Commercialization%20is%20the%20process%20of,the%20new%20product%20or%20service.

[23] (23-May-2020). *Why is Testing Necessary?* Available: https://www.toolsqa.com/software-testing/istqb/why-is-testing-necessary/#:~:text=The%20testing%20is%20important%20since,and%20high%2Dperformance%20software%20operation.

[24] Thanasis. UK Road Safety: Traffic Accidents and Vehicles [Online]. Available: https://www.kaggle.com/tsiaras/uk-road-safety-accidents-and-vehicles#Accident_Information.csv

[25] R. Shaikh, "Feature Selection Techniques in Machine Learning with Python," no. 1-Jan-2020, Available: https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e

[26] S. Ariel, "Accident Severity Classification," no. 12-Jan-2020, Available: https://www.kaggle.com/arielyaakobi10/accident-severity-classification-shirley-ariel

[27] D. Kavyazin, "Principal Component Analysis and k-means Clustering to Visualize a High Dimensional Dataset," Accessed on: 23-June-2020Available: https://medium.com/@dmitriy.kavyazin/principal-component-analysis-and-k-means-clustering-to-visualize-a-high-dimensional-dataset-577b2a7a5fe2

[28] S. Yıldırım, "Decision Trees and Random Forests — Explained," vol. 2020, Available: https://towardsdatascience.com/decision-tree-and-random-forest-explained-8d20ddabc9dd

[29] Z. Little, "AdaBoost," vol. 2020, no. 23-Feb-2020, Available: https://medium.com/@xzz201920/adaboost-7f80fa39584b#:~:text=AdaBoost%20can%20be%20used%20to%20boost%20the%20performance%20of%20any,trees%20on%20binary%20classification%20problems.

[30] R. Khandelwal, "K-Nearest Neighbors(KNN)," Accessed on: 5-May-2020Available: https://medium.com/datadriveninvestor/k-nearest-neighbors-knn-7b4bd0128da7

[31] D. Madushan, "Introduction to K-means Clustering," Accessed on: 23-May-2020Available: https://medium.com/@dilekamadushan/introduction-to-k-means-clustering-7c0ebc997e00#:~:text=The%20K%2Dmeans%20clustering%20algorithm,to%20the%20most%20relevant%20group.

APPENDICES

Appendix A

Source code of the traffic police officers recommendation system

```
    import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
 reason=pd.read_csv('reasons.csv')
officers=pd.read_csv('officers.csv')
officers=pd.read_csv('officers.csv')
```

```
combine_officers_rate = pd.merge(officer, reason, on='officer_id')
combine_officers_rate.isnull()
combine_officers_rate.info()

#implementing KNN
officers_reasons_experience_rating=combine_officers_rate.pivot_table(index
='officer_id',columns='reason',values='experience_rate').fillna(0)
#convert the pivot table into an array matrix
from scipy.sparse import csr_matrix
officers_reasons_matrix = csr_matrix(officers_reasons_experience_rating.va
lues)

#then import NearestNeighbors (unsupervised machine learning)
from sklearn.neighbors import NearestNeighbors

model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model_knn.fit(officers_reasons_matrix)
#default number of n_neighbors=5
#default p (euclidian distance = 2)

#read road accident prediction results
accident_prediction=pd.read_csv('accident_prediction.csv')
#create arrays for each timezone and append the accident percentage
one=[]
two=[]
three=[]
four=[]
#one.append(accident_prediction['time_zone'][row])
for row in accident_prediction.index:
    if accident_prediction['time_zone'][row]==1:
        one.append(accident_prediction['drunk'][row])
        one.append(accident_prediction['careless_driving'][row])
        one.append(accident_prediction['overtake'][row])
        one.append(accident_prediction['rule_violation'][row])
        one.append(accident_prediction['speed'][row])
        one.append(accident_prediction['turning'][row])
        one.append(accident_prediction['other'][row])
    if accident_prediction['time_zone'][row]==2:
        two.append(accident_prediction['drunk'][row])
        two.append(accident_prediction['careless_driving'][row])
        two.append(accident_prediction['overtake'][row])
        two.append(accident_prediction['rule_violation'][row])
        two.append(accident_prediction['speed'][row])
        two.append(accident_prediction['turning'][row])
        two.append(accident_prediction['other'][row])
    if accident_prediction['time_zone'][row]==3:
        three.append(accident_prediction['drunk'][row])
        three.append(accident_prediction['careless_driving'][row])
        three.append(accident_prediction['overtake'][row])
        three.append(accident_prediction['rule_violation'][row])
        three.append(accident_prediction['speed'][row])
        three.append(accident_prediction['turning'][row])
        three.append(accident_prediction['other'][row])
    if accident_prediction['time_zone'][row]==4:
        four.append(accident_prediction['drunk'][row])
        four.append(accident_prediction['careless_driving'][row])
        four.append(accident_prediction['overtake'][row])
        four.append(accident_prediction['rule_violation'][row])
        four.append(accident_prediction['speed'][row])
        four.append(accident_prediction['turning'][row])
        four.append(accident_prediction['other'][row])

print(one,two,three,four)

#get the maximum percentage accident reasons for each time zone
# i=0 -> drunk
# i=1 -> careless_driving
# i=2 -> overtake
# i=3 -> rule_violation
# i=4 -> speed
# i=5 -> turning
# i=6 -> other

#for timezone 1
tm_1 = []

for i in range(0,3):
    f=np.array([one])
    a=f.argmax()
    one[a]=0
    tm_1.append(a)

print(tm_1)
print(one)
```

```
#for timezone 2
tm_2 = []

for i in range(0,3):
    f=np.array([two])
    a=f.argmax()
    two[a]=0
    tm_2.append(a)

print(tm_2)
print(two)

#for timezone 3
tm_3 = []

for i in range(0,3):
    f=np.array([three])
    a=f.argmax()
    three[a]=0
    tm_3.append(a)

print(tm_3)
print(three)

#for timezone 4
tm_4 = []

for i in range(0,3):
    f=np.array([four])
    a=f.argmax()
    four[a]=0
    tm_4.append(a)

print(tm_4)
print(four)

#selecting random officer for timezone_1
first_1=""
second_1=""
third_1=""
j=0
```

```
for i in range(0,8):
    if i==0:
        if i==tm_1[0]:
            first_1='drunk'
        if i==tm_1[1]:
            second_1='drunk'
        if i==tm_1[2]:
            third_1='drunk'
    elif i==1:
        if i==tm_1[0]:
            first_1='careless_driving'
        if i==tm_1[1]:
            second_1='careless_driving'
        if i==tm_1[2]:
            third_1='careless_driving'
    elif i==2:
        if i==tm_1[0]:
            first_1='overtake'
        if i==tm_1[1]:
            second_1='overtake'
        if i==tm_1[2]:
            third_1='overtake'
    elif i==3:
        if i==tm_1[0]:
            first_1='rule_violating'
        if i==tm_1[1]:
            second_1='rule_violating'
        if i==tm_1[2]:
            third_1='rule_violating'
    elif i==4:
        if i==tm_1[0]:
            first_1='speed'
        if i==tm_1[1]:
            second_1='speed'
        if i==tm_1[2]:
            third_1='speed'
    elif i==5:
        if i==tm_1[0]:
            first_1='turning'
        if i==tm_1[1]:
            second_1='turning'
        if i==tm_1[2]:
            third_1='turning'
    else:
        if i==tm_1[0]:
            first_1='other'
        if i==tm_1[1]:
            second_1='other'
        if i==tm_1[2]:
            third_1='other'

print("first : "+ first_1)
print("second : "+ second_1)
print("third : "+ third_1)

off1=[]
off1=np.where((officers_reasons_experience_rating[first_1]==3) & (officers
_reasons_experience_rating[second_1]==3) & (officers_reasons_experience_ra
ting[third_1]==3))
arr=np.array([off1])

result=arr.flatten()
print(result)

random_officer_one=np.random.choice(result)
print(random_officer_one)

#selecting random officer for timezone_2
first_2=""
second_2=""
third_2=""
j=0
for i in range(0,8):
    if i==0:
        if i==tm_2[0]:
            first_2='drunk'
        if i==tm_2[1]:
            second_2='drunk'
        if i==tm_2[2]:
            third_2='drunk'
    elif i==1:
        if i==tm_2[0]:
            first_2='careless_driving'
        if i==tm_2[1]:
            second_2='careless_driving'
        if i==tm_2[2]:
            third_2='careless_driving'
    elif i==2:
        if i==tm_2[0]:
            first_2='overtake'
        if i==tm_2[1]:
            second_2='overtake'
        if i==tm_2[2]:
            third_2='overtake'
    elif i==3:
        if i==tm_2[0]:
            first_2='rule_violating'
        if i==tm_2[1]:
            second_2='rule_violating'
        if i==tm_2[2]:
            third_2='rule_violating'
    elif i==4:
        if i==tm_2[0]:
            first_2='speed'
        if i==tm_2[1]:
            second_2='speed'
        if i==tm_2[2]:
            third_2='speed'
    elif i==5:
        if i==tm_1[0]:
            first_1='turning'
        if i==tm_1[1]:
            second_1='turning'
        if i==tm_1[2]:
            third_1='turning'
    else:
        if i==tm_2[0]:
            first_2='other'
        if i==tm_2[1]:
            second_2='other'
        if i==tm_2[2]:
            third_2='other'

print("first : "+ first_2)
print("second : "+ second_2)
print("third : "+ third_2)
```

```
off2=[]
off2=np.where((officers_reasons_experience_rating[first_2]==3) & (officers
_reasons_experience_rating[second_2]==3) & (officers_reasons_experience_ra
ting[third_2]==3))
arr=np.array([off2])

result=arr.flatten()
print(result)

random_officer_two=np.random.choice(result)
print(random_officer_two)

#selecting random officer for timezone_3
first_3=""
second_3=""
third_3=""
j=0
for i in range(0,8):
    if i==0:
        if i==tm_3[0]:
            first_3='drunk'
        if i==tm_3[1]:
            second_3='drunk'
        if i==tm_3[2]:
            third_3='drunk'
    elif i==1:
        if i==tm_3[0]:
            first_3='careless_driving'
        if i==tm_3[1]:
            second_3='careless_driving'
        if i==tm_3[2]:
            third_3='careless_driving'
    elif i==2:
        if i==tm_3[0]:
            first_3='overtake'
        if i==tm_3[1]:
            second_3='overtake'
        if i==tm_3[2]:
            third_3='overtake'
    elif i==3:
        if i==tm_3[0]:
            first_3='rule_violating'
        if i==tm_3[1]:
            second_3='rule_violating'
        if i==tm_3[2]:
            third_3='rule_violating'
    elif i==4:
        if i==tm_3[0]:
            first_3='speed'
        if i==tm_3[1]:
            second_3='speed'
        if i==tm_3[2]:
            third_3='speed'
    elif i==5:
        if i==tm_3[0]:
            first_3='turning'
        if i==tm_3[1]:
            second_3='turning'
        if i==tm_3[2]:
            third_3='turning'
    else:
        if i==tm_3[0]:
            first_3='other'
        if i==tm_3[1]:
            second_3='other'
        if i==tm_3[2]:
            third_3='other'

print("first : "+ first_3)
print("second : "+ second_3)
print("third : "+ third_3)

off3=[]
off3=np.where((officers_reasons_experience_rating[first_3]==3) & (officers
_reasons_experience_rating[second_3]==3) & (officers_reasons_experience_ra
ting[third_3]==3))
arr=np.array([off3])

result=arr.flatten()
print(result)

random_officer_three=np.random.choice(result)
print(random_officer_three)
```

```
#selecting random officer for timezone_4
first_4=""
second_4=""
third_4=""
j=0
for i in range(0,8):
    if i==0:
        if i==tm_4[0]:
            first_4='drunk'
        if i==tm_4[1]:
            second_4='drunk'
        if i==tm_4[2]:
            third_4='drunk'
    elif i==1:
        if i==tm_4[0]:
            first_4='careless_driving'
        if i==tm_4[1]:
            second_4='careless_driving'
        if i==tm_4[2]:
            third_4='careless_driving'
    elif i==2:
        if i==tm_4[0]:
            first_4='overtake'
        if i==tm_4[1]:
            second_4='overtake'
        if i==tm_4[2]:
            third_4='overtake'
    elif i==3:
        if i==tm_4[0]:
            first_4='rule_violating'
        if i==tm_4[1]:
            second_4='rule_violating'
        if i==tm_4[2]:
            third_4='rule_violating'
    elif i==4:
        if i==tm_4[0]:
            first_4='speed'
        if i==tm_4[1]:
            second_4='speed'
        if i==tm_4[2]:
            third_4='speed'
    elif i==5:
        if i==tm_4[0]:
            first_4='turning'
        if i==tm_4[1]:
            second_4='turning'
        if i==tm_4[2]:
            third_4='turning'
    else:
        if i==tm_4[0]:
            first_4='other'
        if i==tm_4[1]:
            second_4='other'
        if i==tm_4[2]:
            third_4='other'

print("first : "+ first_4)
print("second : "+ second_4)
print("third : "+ third_4)

off4=[]
# off4=np.where((test_reasons_experience_rating[first_4]==1) & (test_reaso
ns_experience_rating[second_4]==1) & (test_reasons_experience_rating[third
_4]==1))
off4=np.where((officers_reasons_experience_rating[first_4]==3) & (officers
_reasons_experience_rating[second_4]==3) & (officers_reasons_experience_ra
ting[third_4]==3))
arr=np.array([off4])

result=arr.flatten()
print(result)

random_officer_four=np.random.choice(result)
print(random_officer_four)

flattened = pd.DataFrame(officers_reasons_experience_rating.to_records())

# test_reasons_experience_rating
officer_one=[]
distances_one, indices_one = model_knn.kneighbors(officers_reasons_experie
nce_rating.iloc[random_officer_one,:].values.reshape(1, -
1), n_neighbors = 21)
indices_one
```

```
print("recommended police officers for time_zone one")
for i in range(0, len(distances_one.flatten())):

    print('{0}: {1}, with distance of {2}:'.format(i, officers_reasons_exp
erience_rating.index[indices_one.flatten()[i]], indices_one.flatten()[i]))
    officer_one.append(officers_reasons_experience_rating.index[indices_on
e.flatten()[i]])

officer_one

jsonOffOne.to_json(r'C:\Users\User\Documents\police_scheduling\officers_da
ta\time_zone_one\officers1.json')

#recommending officers for timezone 2
# test_reasons_experience_rating
officer_two=[]
distances_one, indices_one = model_knn.kneighbors(officers_reasons_experie
nce_rating.iloc[random_officer_two,:].values.reshape(1, -
1), n_neighbors = 21)
indices_one
print("recommended police officers for time_zone two")
for i in range(0, len(distances_one.flatten())):
    print('{0}: {1}, with distance of {2}:'.format(i, officers_reasons_exp
erience_rating.index[indices_one.flatten()[i]], indices_one.flatten()[i]))
    officer_two.append(officers_reasons_experience_rating.index[indices_on
e.flatten()[i]])

#officer_two
jsonOffTwo = DataFrame(officer_two, columns=['Officers'])
jsonOffTwo

jsonOffTwo.to_json(r'C:\Users\User\Documents\police_scheduling\officers_da
ta\time_zone_two\officers2.json')


#recommending officers for timezone 4
# test_reasons_experience_rating
officer_three=[]
distances_one, indices_one = model_knn.kneighbors(officers_reasons_experie
nce_rating.iloc[random_officer_three,:].values.reshape(1, -
1), n_neighbors = 21)
indices_one
```

```
print("recommended police officers for time_zone three")
for i in range(0, len(distances_one.flatten())):
    print('{0}: {1}, with distance of {2}:'.format(i, officers_reasons_exp
erience_rating.index[indices_one.flatten()[i]], indices_one.flatten()[i]))
    officer_three.append(officers_reasons_experience_rating.index[indices_
one.flatten()[i]])

print(officer_three)

#officer_three
jsonOffThree = DataFrame(officer_three, columns=['Officers'])
jsonOffThree.to_json(r'C:\Users\User\Documents\police_scheduling\officers_
data\time_zone_three\officers3.json')

#recommending officers for timezone 4
# test_reasons_experience_rating
officer_four=[]
distances_one, indices_one = model_knn.kneighbors(officers_reasons_experie
nce_rating.iloc[random_officer_four,:].values.reshape(1, -
1), n_neighbors = 21)
indices_one
print("recommended police officers for time_zone four")
for i in range(0, len(distances_one.flatten())):
    print('{0}: {1}, with distance of {2}:'.format(i, officers_reasons_exp
erience_rating.index[indices_one.flatten()[i]], indices_one.flatten()[i]))
    officer_four.append(officers_reasons_experience_rating.index[indices_o
ne.flatten()[i]])

print(officer_four)

jsonOffFour = DataFrame(officer_four, columns=['Officers'])
jsonOffFour.to_json(r'C:\Users\User\Documents\police_scheduling\officers_d
ata\time_zone_four\officers4.json')
```

Source code of the traffic police officers scheduling

```
import pandas as pd
```

```
officerOne = pd.read_json(r'C:\Users\User\Documents\police_scheduling\offi
cers_data\time_zone_one\officers1.json')
officerTwo = pd.read_json(r'C:\Users\User\Documents\police_scheduling\offi
cers_data\time_zone_two\officers2.json')
officerThree = pd.read_json(r'C:\Users\User\Documents\police_scheduling\of
ficers_data\time_zone_three\officers3.json')
officerFour = pd.read_json(r'C:\Users\User\Documents\police_scheduling\off
icers_data\time_zone_four\officers4.json')
from gurobipy import *
officersTZOne=officerOne.values.tolist()
officersTZOne = [x[0] for x in officersTZOne]

officersTZTwo=officerTwo.values.tolist()
officersTZTwo = [x[0] for x in officersTZTwo]

officersTZThree=officerThree.values.tolist()
officersTZThree = [x[0] for x in officersTZThree]

officersTZFour=officerFour.values.tolist()
officersTZFour = [x[0] for x in officersTZFour]

#lists
M = ["chkPoint1","chkPoint2","chkPoint3","chkPoint4","chkPoint5"]
D = ["Mon","Teu","Wed","Thur","Fri","Sat","Sun"]

#parameters
Hm = 3

#constants
S = 5;

from gurobipy import *

modelOne=Model("time_zone_one")
modelTwo=Model("time_zone_two")
modelThree=Model("time_zone_three")
modelFour=Model("time_zone_four")

zone=modelOne.addVars(officersTZOne,M,D, vtype=GRB.BINARY, name="zone")
N=len(officersTZOne)
possibilities =range(N*3)
possibilities
```

```
modelOne.setObjective(
quicksum([zone[n,m,d] for n in officersTZOne for m in M for d in D]), sens
e=GRB.MINIMIZE)

modelOne.addConstrs((quicksum([zone[(n,m,d)] for m in M for d in D])<=S fo
r n in officersTZOne), name="maximum_shifts_per_officer");
modelOne.addConstrs((quicksum([zone[(n,m,d)] for n in officersTZOne])>=Hm
for m in M for d in D), name="minimum_officers_for_each_checkpoin");
modelOne.addConstrs((quicksum([zone[(n,m,d)] for n in officersTZOne for m
in M for d in D])==(N*S) for p in possibilities), name="all_officers_in_al
l_checkpoints");
#In one particulat day a police officer can be allocated only in one or no
 checkpoint. otherwise one police officer might get scheduled in more tha
1 checkpoint in a particular day which is not possible.
modelOne.addConstrs((quicksum([zone[(n,m,d)] for m in M])<=1 for n in offi
cersTZOne for d in D),name="One officer at one checkpoint at any given day
");

modelOne.optimize()

modelOne.printAttr('X')

ztwo=modelTwo.addVars(officersTZTwo,M,D, vtype=GRB.BINARY, name="ztwo")

N=len(officersTZTwo)
possibilities =range(N*3)
possibilities

modelTwo.setObjective(
quicksum([ztwo[n,m,d] for n in officersTZTwo for m in M for d in D]), sens
e=GRB.MINIMIZE)

modelTwo.setObjective(
quicksum([ztwo[n,m,d] for n in officersTZTwo for m in M for d in D]), sens
e=GRB.MINIMIZE)
modelTwo.addConstrs((quicksum([ztwo[(n,m,d)] for n in officersTZTwo])>=3 f
or m in M for d in D), name="minimum_officers_for_each_checkpoin");
modelTwo.addConstrs((quicksum([ztwo[(n,m,d)] for n in officersTZTwo for m
in M for d in D])==(N*S) for p in possibilities), name="all_officers_in_al
l_checkpoints");
```

```
modelTwo.addConstrs((quicksum([ztwo[(n,m,d)] for m in M])<=1 for n in offi
cersTZTwo for d in D),name="One officer at one checkpoint at any given day
")

modelTwo.optimize()

modelTwo.printAttr('X')

zthree=modelThree.addVars(officersTZThree,M,D, vtype=GRB.BINARY, name="zth
ree")

N=len(officersTZThree)
possibilities =range(N*3)
possibilities


modelThree.setObjective(
quicksum([zthree[n,m,d] for n in officersTZThree for m in M for d in D]),
sense=GRB.MINIMIZE)

modelThree.addConstrs((quicksum([zthree[(n,m,d)] for m in M for d in D])<=
S for n in officersTZThree), name="maximum_shifts_per_officer");
modelThree.addConstrs((quicksum([zthree[(n,m,d)] for n in officersTZThree]
)>=3 for m in M for d in D), name="minimum_officers_for_each_checkpoin");
modelThree.addConstrs((quicksum([zthree[(n,m,d)] for n in officersTZThree
for m in M for d in D])==(N*S) for p in possibilities), name="all_officers
_in_all_checkpoints");
modelThree.addConstrs((quicksum([zthree[(n,m,d)] for m in M])<=1 for n in
officersTZThree for d in D),name="One officer at one checkpoint at any giv
en day")

modelThree.optimize()

modelThree.printAttr('X')

zfour=modelFour.addVars(officersTZFour,M,D, vtype=GRB.BINARY, name="zfour"
)
N=len(officersTZFour)
possibilities =range(N*3)
possibilities

modelFour.setObjective(
quicksum([zfour[n,m,d] for n in officersTZFour for m in M for d in D]), se
nse=GRB.MINIMIZE)
modelFour.addConstrs((quicksum([zfour[(n,m,d)] for m in M for d in D])<=S
for n in officersTZFour), name="maximum_shifts_per_officer");
modelFour.addConstrs((quicksum([zfour[(n,m,d)] for n in officersTZFour])>=
3 for m in M for d in D), name="minimum_officers_for_each_checkpoin");
modelFour.addConstrs((quicksum([zfour[(n,m,d)] for n in officersTZFour for
 m in M for d in D])==(N*S) for p in possibilities), name="all_officers_in
_all_checkpoints");
modelFour.addConstrs((quicksum([zfour[(n,m,d)] for m in M])<=1 for n in of
ficersTZFour for d in D),name="One officer at one checkpoint at any given
day")

modelFour.optimize()

modelFour.printAttr('X')
```