1. INTRODUCTION

The popularization of vehicles in our daily life has been continuously enhanced with the expansion of urbanization around the world. Gasoline-engine vehicles are the most popular and widely used type compared with new energy ones, and the pollution gases, such as carbon dioxide, carbon oxide, hydrocarbon, and oxynitride, from vehicles have become the main contaminants in urban atmospheric pollution [1]. Efficient vehicle pollution detection therefore turns to be an emergency task which attracks more and more attention. Exhaust emission detection methods have evolved from periodic detection in the environmental monitoring station to daily road detection with remote sensing technology. This paper studies the vehicle emission detection in cities of China which is one of the largest developing countries.

In the USA, EPA (Environmental Protection Administration) proposed MOVES algorithm [2] to calculate the vehicle emission ratio in some fixed locations and periods of time. The Japanese government enforces the vehicle exhaust emission monitoring system in their country, and the emission behaviour of each vehicle in Japan can be checked on the official website of Japanese national transportation [3]. In order to rapidly capture the emission detection results, a French transport agency collects the emission pollution-related information from different places and puts them together to realize the sharing network for vehicle emission detection [4]. Related researches and works on this area started a bit later in China. In 2011, Cheng et al. [5] made systematic analysis for the harm caused by vehicle emission, verifying the necessities of exhaust emission controlling. Next year, Wu [6] collected the values of CO2, HC, CO, and NO exhausted by 1092 vehicles in the Xian Yang city using simplified loaded mode. They established regression equations between the emission value and vehicle information and found that the average emission value was highly related with the vehicle acceptability and the age of the vehicle. Referring to the local standards, they further gave a systematic explanation for the rationalization of the local standard mean emission value based on their research. With the development of remote sensing technology, a large amount of practical exhaust emission data can be obtained by environmental protection agencies in China. This paper introduces data mining technology to these valuable data to explore efficient information in vehicle exhaust emission detection. This research has a huge potential contribution in promoting

the environmental protection department's accurate assessment of unqualified vehicles and providing a theoretical basis for policymakers to learn from.

The first successful vehicle emissions demonstration system was probably an across-road vehicle emissions remote sensing system (VERSS) proposed by Gary Bishop and colleagues in the University of Denver in the late 1980s [7, 8]. A liquid nitrogen cooled nondispersive infrared was the first instrument that can only measure CO and CO2. In the next two decades, their team continuously refined the system: added hydrocarbon, H2O, and NO channels to their NDIR system [9, 10], integrated an ultraviolet spectrophotometer and improved it to enhance NO measurement [11, 12], and removed the dependence on the liquid nitrogen cooling [13]. The Denver group designed another commonly used remote sensing device, known as fuel efficiency automobile test, providing some of the inchoate comments on across-road particulate measurement [14]. There are also many other sensing systems typically based on multiple spectrometric approaches proposed for detection of passing vehicle emissions [15–17]. More recently, Hager Environmental and Atmospheric Technologies introduced an infrared laser-based VERSS named Emission Detection and Reporting (EDAR) system, which incorporated several new functions, making it a particularly interesting system for vehicle emission detection.

Important information is buried in the vehicle emission remote sensing data. This paper exploits data mining methods to deal with the data and obtain valuable knowledge from them. There are three main directions in data mining: the improvements of classical data mining algorithms, ensemble learning algorithms, and data mining with deep learning. The improvements on classical algorithms are usually performed and employed in multiple application scenarios taking additional information into consideration. Ensemble learning is actually the integration of multiple learners with a certain structure which completes learning tasks by constructing and combining different learners. Its general structure can be concluded as follows: firstly, generate a set of individual learners and then combine them with some strategies. The combining strategies mainly include average method, voting method, and learning method. Bagging and boosting [18] are the most commonly used ensemble learning algorithms which improve the accuracy and robustness of prediction models. As the rapid development and popularization of deep learning, it plays more and more important roles in data learning with the support of big

data and high-performance computing. Many traffic engineering-related researches mainly focus on analyzing relevant data such as traffic diversion [19], traffic safety monitoring [20], engine diagnosis [21], road safety [22] and traffic accident [23], and remote sensing image processing [24–35], extracting useful information and digging out valuable knowledge. A few works are proposed in vehicle emission evaluation in data mining ways which is the key study subject in this paper. Xu et al. [36] used XgBoost to develop prediction models for CO2eq and PM2.5 emissions at a trip level. In [37], Ferreira et al. applied online analytical processing (OLAP) and knowledge discovery (KD) techniques to deal with the high volume of this dataset and to determine the major factors that influence the average fuel consumption and then classify the drivers involved according to their driving efficiency. Chen et al. [38] proposed a driving-events-based ecodriving behaviour evaluation model and the model was proved to be highly accurate (96.72%).

Relevant environmental policies have been introduced to define difficult limitation standards based on the vehicle fuel type and registration time in China. The vehicle license plate number, plate color, speed, acceleration, and VSP (vehicle specific power), etc., will be captured by the surveillance system when vehicles pass by the remote survey stations. The analysis for the smoke plume generated by gas emission is simultaneously conducted by laser gears at the stations, where the exhaust emission value can be calculated. With the fuel type and registration time information learned from vehicle plate numbers, it is able to obtain the gas emission standard value to judge whether the vehicle emission is eligible. However, register information of nonlocal vehicles and partial local vehicles is not recorded in the official database due to the limitation of environmental policies, which leads to the failure to provide the fuel type and registration time information for vehicle emission detection. According to the National Telemetry Standard in China, relevant departments will treat the information-missing vehicles as the diesel consumption ones, and this situation keeps the limitation criteria of the emission value of partial vehicles unknown, resulting in the evaluation for these vehicles being unable to carry on. Therefore, the precise information upon fuel types and registration time of vehicles is an essential prerequisite for finding out the pollution-exceeding vehicles. This paper adopts multiple data mining methods to learn the fuel type and registration information of vehicles from remote sensing data and further utilize cascaded classified framework to make accurate prediction on vehicle emission-related information, providing valuable reference standards on evaluation of different vehicles.

1.1 Problem statement

With the rise in urbanization and vehicular usage, air pollution has become a critical environmental concern worldwide. A significant contributor to this pollution is the emission of harmful gases from vehicles, especially older or poorly maintained ones. Traditional emission testing methods are often manual, infrequent, and unable to provide real-time insights into a vehicle's environmental impact. There is a growing need for an intelligent system that can predict and evaluate vehicle emissions using historical data, helping users understand whether their vehicle is eco-friendly or contributing heavily to pollution.

This project aims to develop a web-based vehicle emission detection and alert system using machine learning models. The system analyzes various input parameters such as engine size, fuel consumption, vehicle make, and type of fuel to predict the CO2 emission of a vehicle. Based on the predicted emission level, the system alerts the user if their vehicle exceeds a predefined environmental threshold, encouraging proactive maintenance or replacement to reduce environmental impact.

1.2 Motivation

The alarming rise in air pollution, particularly in urban areas, is a pressing global issue with serious health and environmental consequences. One of the major contributors to this problem is the increasing number of vehicles on the road emitting high levels of carbon dioxide (CO2) and other pollutants. Despite regulations, many vehicles continue to emit beyond permissible limits due to inadequate maintenance, lack of awareness, or outdated testing methods.

This situation inspired the development of a smart, data-driven solution that can help individuals and policymakers monitor and control vehicle emissions. By leveraging machine learning algorithms and real-time data analysis, this project offers a predictive system that can assess a vehicle's emission level based on key parameters. The motivation is to create a user-friendly tool that not only detects high-emission vehicles but also encourages users to make informed decisions for reducing their environmental impact. This proactive approach contributes to cleaner air, better public health, and sustainable urban living.

1.3 Objective

Develop a comprehensive system utilizing Air Quality Detection technology to continuously monitor and detect vehicle emissions on roads. Identify vehicles that exceed emission limits and implement an Instant Owner Notification mechanism to promptly inform car owners about their emissions' status. Encourage responsible driving behaviour and emission reduction among car owners through timely notifications, creating awareness about their carbon footprint. Introduce a Progressive Warning System with escalating consequences for non-compliance, incentivizing car owners to take necessary actions to reduce their emissions.

1.4 Existing System

Environmental protection is a fundamental policy in many countries, where the vehicle emission pollution turns to be outstanding as a main component of pollutions in environmental monitoring. Remote sensing technology has been widely used on vehicle emission detection recently and this is mainly due to the fast speed, reality, and large scale of the detection data retrieved from remote sensing methods. In the remote sensing process, the information about the fuel type and registration time of new cars and nonlocal registered vehicles usually cannot be accessed, leading to the failure in assessing vehicle pollution situations directly by analyzing emission pollutants.

Disadvantages

- Manual analysis will not give better prediction
- No machine learning algorithm were used

1.5 Proposed System

This paper adopts data mining methods to analyze the Vehicle emission data to vehicle emission. This paper takes full use of linear regression, random forest, KNN, XgBoost, to successfully make precise prediction for essential information and further employ them to an essential application: vehicle emission evaluation.

Advantages

- We are using machine learning algorithms to analysis the data
- Analyzing with multiple ML algorithms
- Gives better prediction

2. REQUIREMENT ANALYSIS

2.1 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

2.1.1 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.1.2 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.1.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includesthe process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of

acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.2 Software Requirements Specifications

2.2.1 Hardware Requirements

System : Pentium IV 2.4 GHz.

Hard Disk : 40 GB.

Ram : 512 Mb.

2.2.2 Software Requirements

Operating system : Windows.

Coding Language : Python.

Designing : Html, css.

Framework : django

3. TECHNOLOGY USED

3.1 Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Advantages of Python :-

Let's see how Python dominates over other languages.

Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand,and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with codereusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

Free And Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

- **Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

- **Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

- **Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate

things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle. The reason it is not so famous despite the existence of Brython is that it isn'tthat secure.

Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

3.2 Machine learning

What is Machine Learning: -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building model of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameter* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Leaning:-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then

the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning:-

While Machine Learning is rapidly evolving, making significant strides with cyber security and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of over fitting & underfitting − if the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.0

Applications of Machines Learning:-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech recognition
- Object recognition
- Fraud detection
- Fraud prevention

How to Start Learning Deep Learning?

Arthur Samuel coined the term **"Deep Learning"** in 1959 and defined it as a **"Field of study that gives computers the capability to learn without being explicitly programmed".**And that was the beginning of Deep Learning! In modern times, Deep Learning is one of the most popular (if not the most!) career choices. According to Indeed, Deep Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Deep Learning and how to start learning it? So this article deals with the Basics of Deep Learning and also the path you can follow to eventually become a full-fledged Deep Learning Engineer. Now let's get started!!!

How to start learning DL?

This is a rough roadmap you can follow on your way to becoming an insanely talented Deep Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start DL directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Deep Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy Deep Learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Deep Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many DL algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Deep Learning. In fact, around 80% of your time as an DL expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!

Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of DL which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Deep Learning like R, Scala, etc. Python is currently the most popular language for DL. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Deep Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn DL, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various DL Concepts

Now that you are done with the prerequisites, you can move on to actually learning DL (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in DL are:

(a) Terminologies of Deep Learning

Model – A model is a specific representation learned from data by applying some Deep Learning algorithm. A model is also called a hypothesis.

Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

Training – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.

Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Deep Learning

- **Supervised Learning –** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning –** This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Deep Learning:-

- **Easily identifies trends and patterns -**

Deep Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

- **No human intervention needed (automation)**

With DL, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. DL is also good at recognizing spam.

- **Continuous Improvement**

As **DL algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

- **Handling multi-dimensional and multi-variety data**

Deep Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

- **Wide Applications**

You could be an e-tailer or a healthcare provider and make DL work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Deep Learning :-

- **Data Acquisition**

Deep Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

- **Time and Resources**

DL needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

- **Interpretation of Results**

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

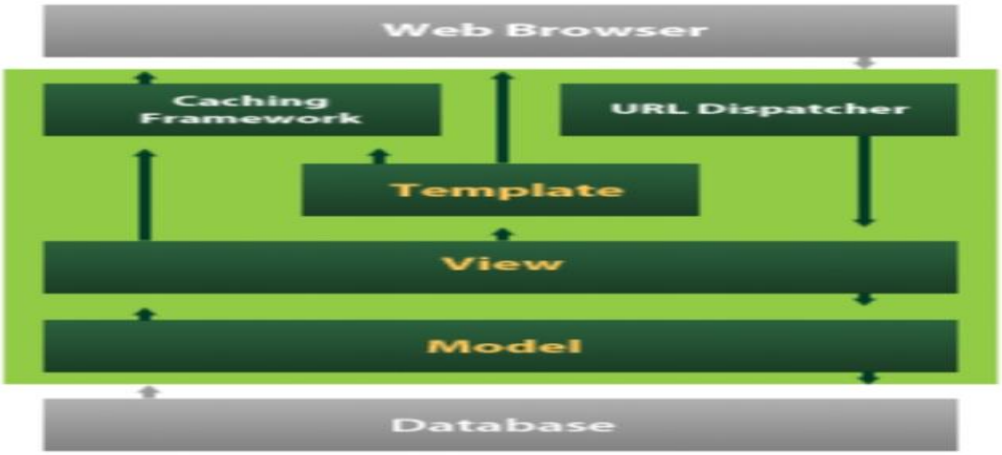
- **High error-susceptibility**

Deep Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of DL, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

3.3 Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusabilityand "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.



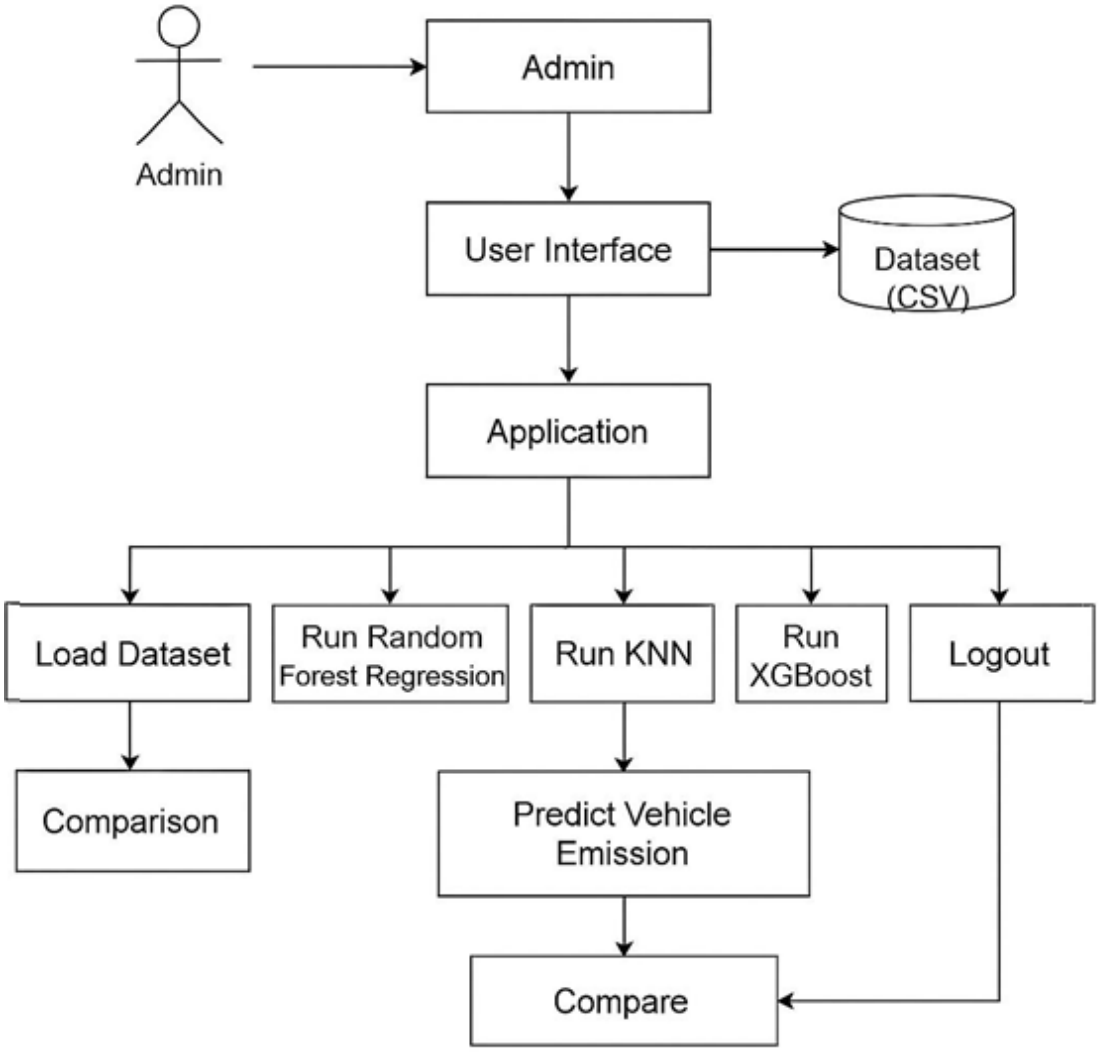
3.3.1.1 Django

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

4. ARCHITECTURE DESIGN

4.1 System Architecture

The system architecture of the Vehicle Emission Detection and Alert project is designed as a modular, layered architecture consisting of several interconnected components that enable data loading, preprocessing, model training, prediction, and result visualization. The system is implemented using a web-based interface built with Django, which interacts with the machine learning backend for performing the core tasks.



4.1.1.1 System Architecture

4.2 Modules used in project

Admin: Here admin is a module should login into the account after successful login admin can perform some operations such as

Load Dataset: Here loading dataset from dataset folder available in project

Pre-process: in this step we are going to clean dataset and defining predicator and target variables and splitting data into training and testing.

Run Linear Regression: here training this algorithm with training dataset which is split in pre-process stage

Run Random Forest Regression: here training this algorithm with training dataset which is split in preprocess stage

Run KNN: here training this algorithm with training dataset which is split in pre-process stage

Run XGBOOST: here training this algorithm with training dataset which is split in preprocess stage

Comparison: showing comparison graph with all algorithm generated accuracy

Predict Vehicle Emission: in this step we need to predict vehicle emission by passing test data to the algorithm

Logout: this operation for coming out from admin accout

4.3 UML Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

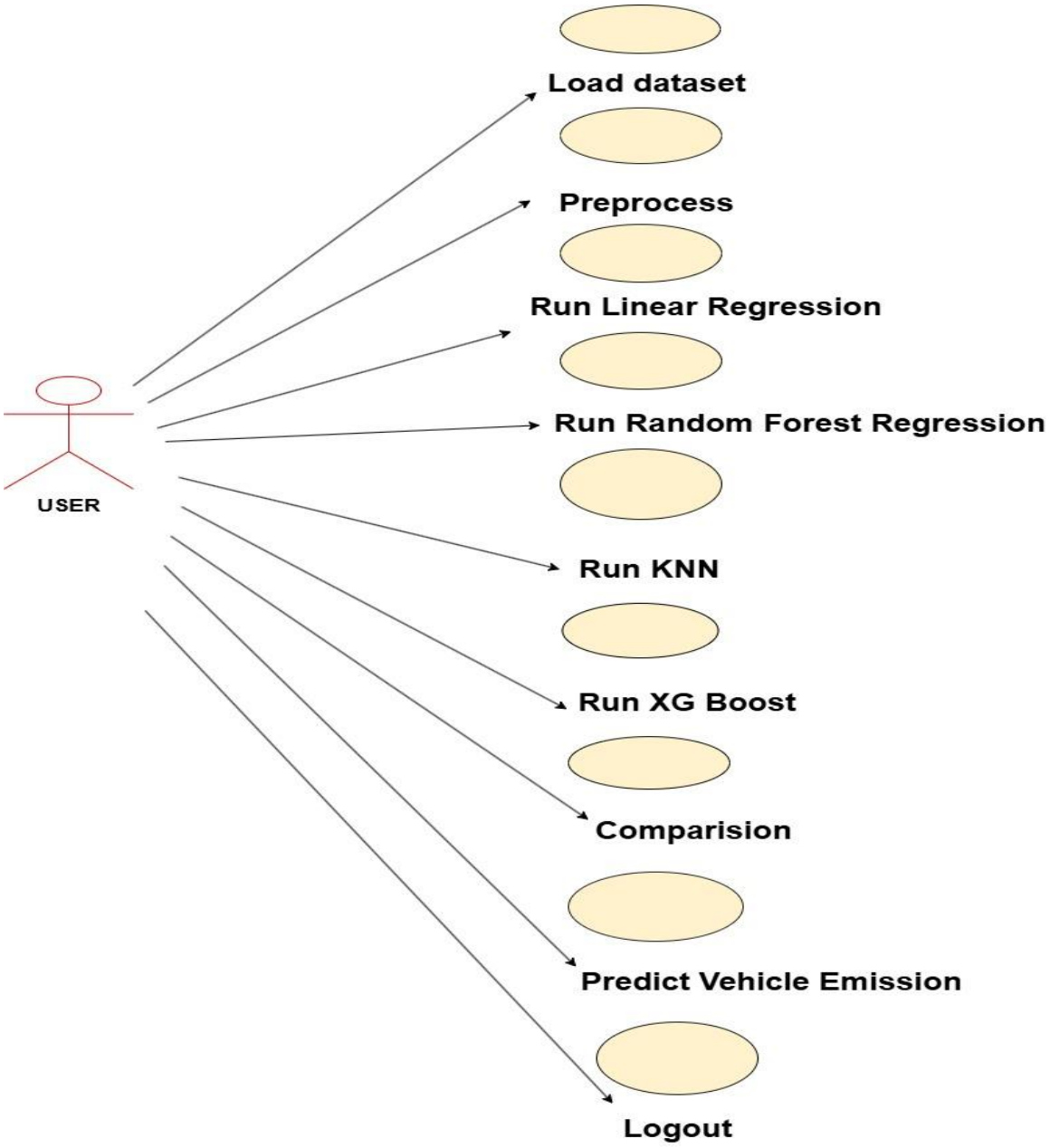
Goals:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

Use Case Diagram

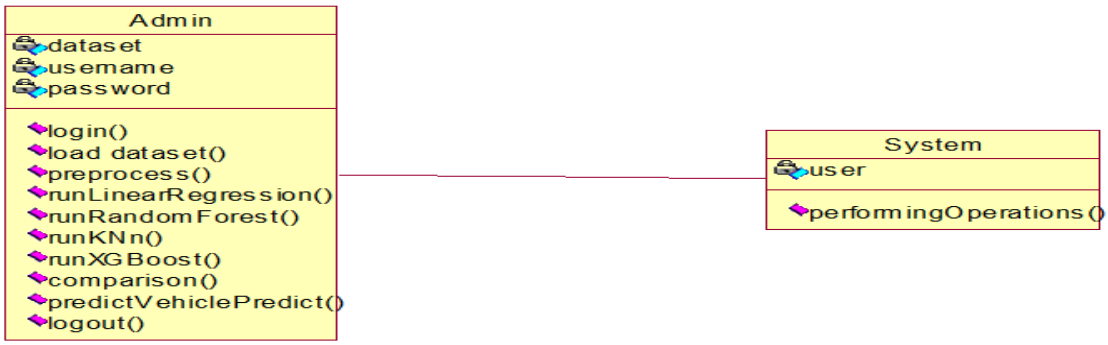
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



4.3.1.1 Use Case Diagram

Class Diagram

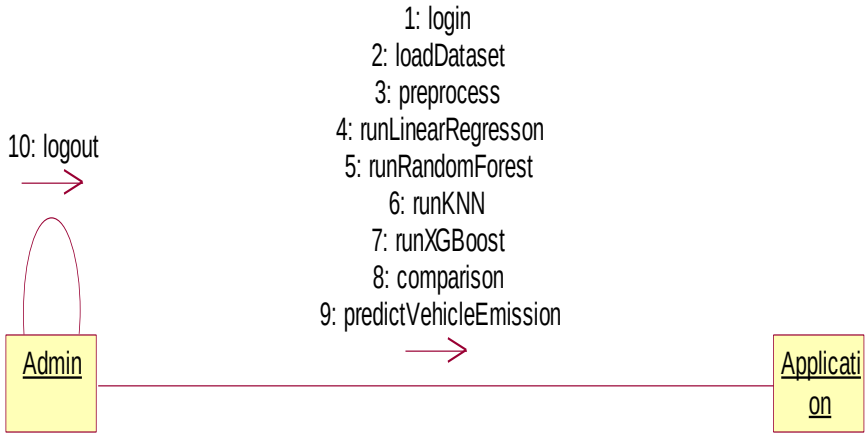
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



4.3.1.2 Class Diagram

Collaboration:

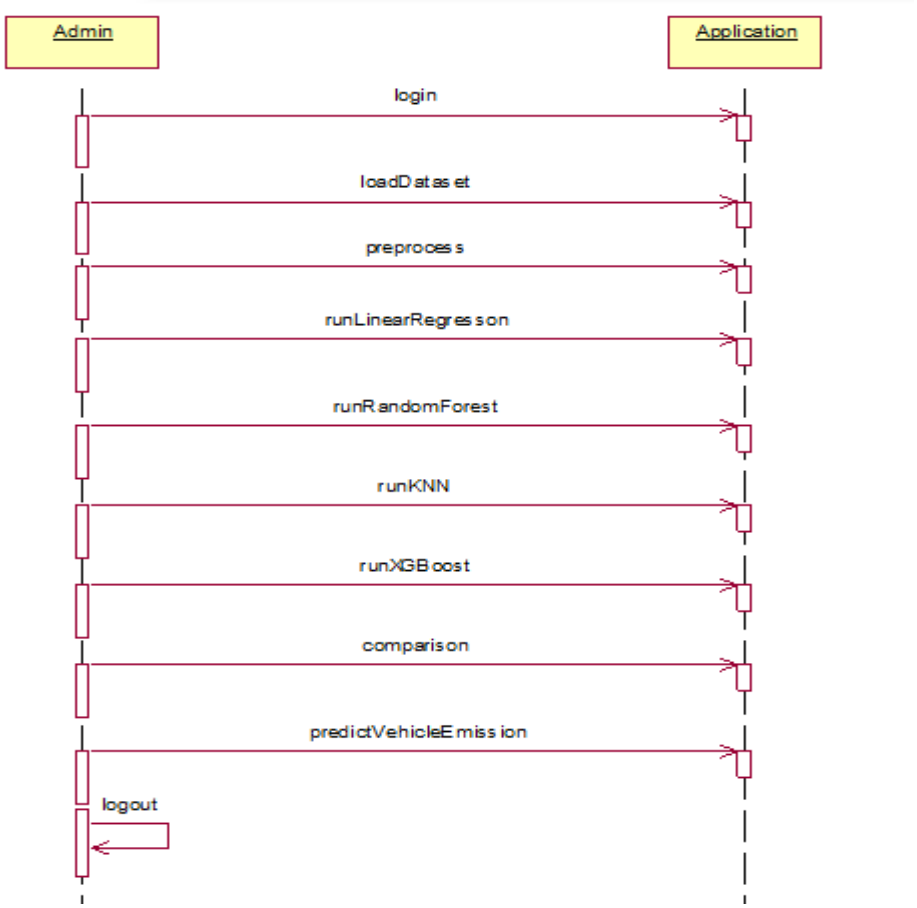
A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.



4.3.1.3 Collaboration Diagram

Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



4.3.1.4 Sequence Diagram

5. IMPLEMENTATION

5.1 Sample Code

```
from django.shortcuts import render

import pymysql

import os

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn import svm

from sklearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.neighbors import KNeighborsRegressor

import xgboost as xg

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

import random



# Create your views here.

def index(request):

    return render(request,'AdminApp/index.html')
```

```
def login(request):
    return render(request,'AdminApp/Admin.html')
def LogAction(request):
    username=request.POST.get('username')
    password=request.POST.get('password')
    if username=='Admin' and password=='Admin':
        return render(request,'AdminApp/AdminHome.html')
    else:
        context={'data':'Login Failed ....!!'}
        return render(request,'AdminApp/Admin.html',context)
def home(request):
    return render(request,'AdminApp/AdminHome.html')
global df
def LoadData(request):
    global df
    BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    df=pd.read_csv(BASE_DIR+"\\dataset\\CO2 Emissions_Canada.csv")
    #data.fillna(0, inplace=True)
    context={'data':"Dataset Loaded\n"}


    return render(request,'AdminApp/AdminHome.html',context)
```

```
global X

global y

global X_train,X_test,y_train,y_test

def split(request):

    global X_train,X_test,y_train,y_test

    global df

    #df1=pd.DataFrame({col: df[col].astype('category').cat.codes for col in df}, index=df.index)

    df['Make']=df['Make'].map({'ACURA':1,'ALFA ROMEO':2,'ASTON MARTIN':3,'AUDI':4,'BENTLEY':5,'BMW':6,'BUICK':7,'CADILLAC':8,'CHEVROLET':9,'CHRYSLER':10,'DODGE':11,'FIAT':12,'FORD':13,'GMC':14,'HONDA':15,'HYUNDAI':16,'INFINITI':17,'JAGUAR':18,'JEEP':19,'KIA':20,'LAMBORGHINI':21,'LAND ROVER':22,'LEXUS':23,'LINCOLN':24,'MASERATI':25,'MAZDA':26,'MERCEDES-BENZ':27,'MINI':28,'MITSUBISHI':29,'NISSAN':30,'PORSCHE':31,'RAM':32,'ROLLS-ROYCE':33,'SCION':34,'SMART':35,'SRT':36,'SUBARU':37,'TOYOTA':38,'VOLKSWAGEN':39,'VOLVO':40,'GENESIS':41,'BUGATTI':42})

    df['Fuel Type']=df['Fuel Type'].map({'Z':1,'D':2,'X':3,'E':4,'N':5})


df['Transmission']=df['Transmission'].map({'AS5':1,'M6':2,'AV7':3,'AS6':4,'AM6':5,'A6':6,'AM7':7,'AV8':8,'AS8':9,'A7':10,'A8':11,'M7':12,'A4':13,'M5':14,'AV':15,'A5':16,'AS7':17,'A9':18,'AS9':19,'AV6':20,'AS4':21, 'AM5':22, 'AM8':23, 'AM9':24, 'AS10':25, 'A10':26, 'AV10':27})

    df=df.drop(columns=['Model','Vehicle Class'])

    X=df[['Make','Engine Size(L)','Cylinders','Transmission','Fuel Type','Fuel Consumption City (L/100 km)','Fuel Consumption Hwy (L/100 km)','Fuel Consumption Comb (L/100 km)','Fuel Consumption Comb (mpg)']]

    y = df['CO2 Emissions(g/km)']
```

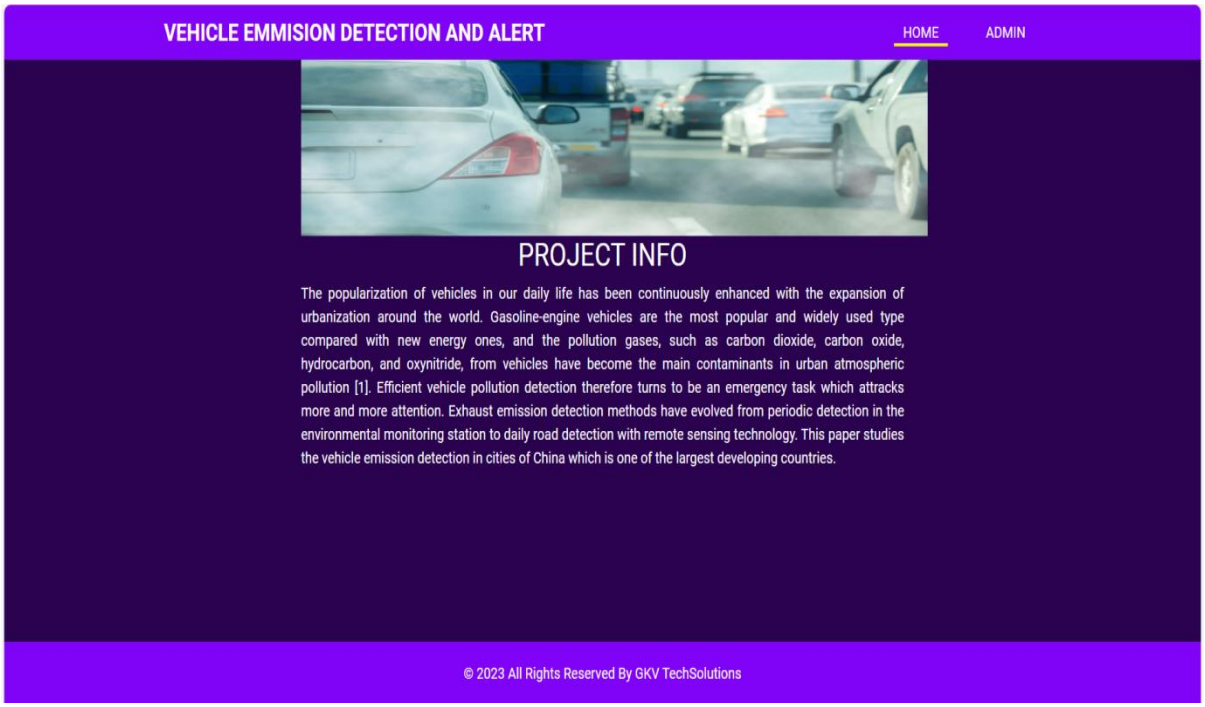
```
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)

    table="<table border='1' style='margin-top:100px;'><tr><th>Total Dataset Records</th><th>80% records as training data</th><th>20% records as test data</th></tr>"


table+="<tr><td>"+str(len(df))+"</td><td>"+str(len(X_train))+"</td><td>"+str(len(y_test))+"</td></tr>"

    table+="</table>"

    context={"data":table,"data2":"DataSet Preprocessed and Splitted Data"}

    return render(request,'AdminApp/AdminHome.html',context)

global LRacc

global LRModel

def runLinearRegression(request):

    global LRacc

    global LRModel

    LRModel = LinearRegression()

    LRModel.fit(X_train, y_train)

    LRacc=LRModel.score(X_train, y_train)*100

    context={"data":"Linear Regression Accurary: "+str(LRacc)}

return render(request,'AdminApp/AdminHome.html',context)

global RRacc

global Rmodel

def runRandomRegression(request):
```

```
    global RRacc
    global Rmodel
    Rmodel = RandomForestRegressor()
    Rmodel.fit(X_train, y_train)
    RRacc=Rmodel.score(X_train, y_train)*100
    context={"data":"RandomForest Accurary: "+str(RRacc)}
    return render(request,'AdminApp/AdminHome.html',context)
global knnacc
global knnmodel
def runKNeighborsRegressor(request):
    global knnacc
    global knnmodel
    knnmodel = KNeighborsRegressor()
    knnmodel.fit(X_train, y_train)
    knnacc=knnmodel.score(X_train, y_train)*100
    context={"data":"KNNeighbors Accurary: "+str(knnacc)}
return render(request,'AdminApp/AdminHome.html',context)
global XGacc
global XGmodel
def runXGBoost(request):
    global XGacc
```

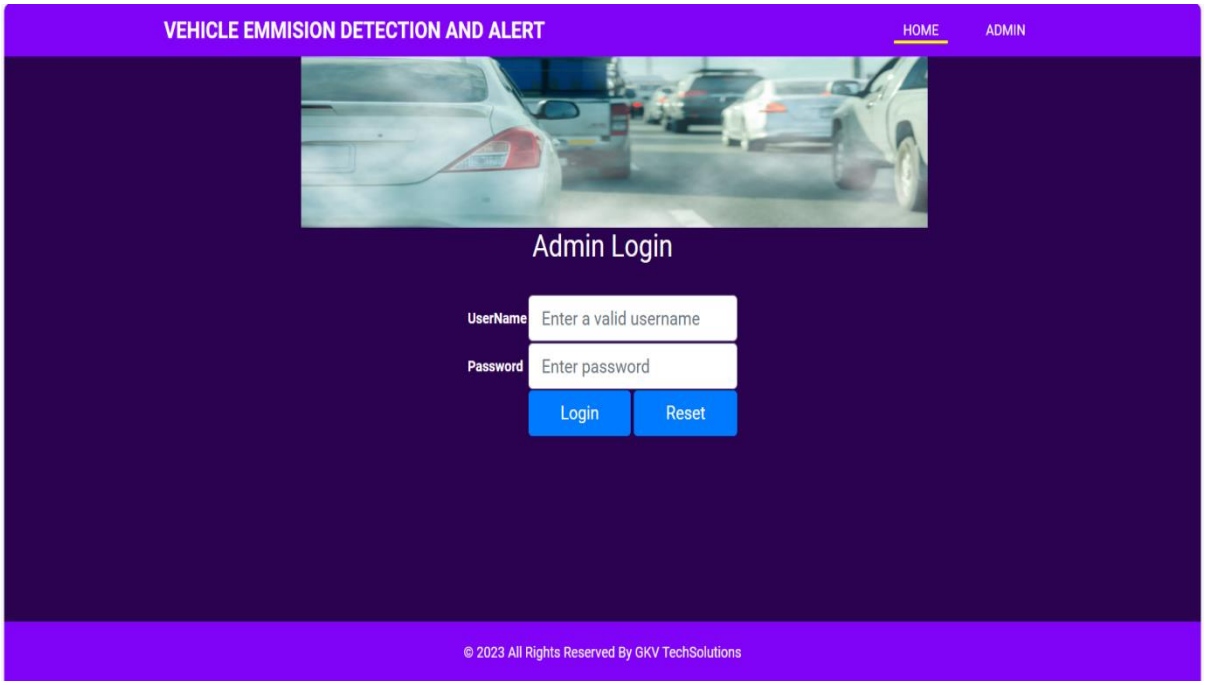
```
    global XGmodel
    XGmodel = xg.XGBRegressor()
    XGmodel.fit(X_train, y_train)
    XGacc=XGmodel.score(X_train, y_train)*100
    context={"data":"XGBoost Accurary: "+str(XGacc)}
    return render(request,'AdminApp/AdminHome.html',context)
  def runComparision(request):
    global LRacc,RRacc,knnacc,XGacc
    bars = ['Linear Regression','RandomforestRegression','KNN Regresion','XGBoost']
    height = [LRacc,RRacc,knnacc,XGacc]
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()
    return render(request,'AdminApp/AdminHome.html')
def predict(request):
    return render(request,'AdminApp/Prediction.html')
def PredAction(request):
    global model
    global Rmodel
    g=request.POST.get('Make')
```

```
    a=request.POST.get('engsize')
    hyp=request.POST.get('cylinders')
    heart=request.POST.get('trans')
    em=request.POST.get('fuel')
    wt=request.POST.get('fuelconsumption')
    Rt=request.POST.get('fchwy')
    agl=request.POST.get('fuelconcom')
    bmi=request.POST.get('fccomb')
  pred=Rmodel.predict([[g,a,hyp,heart,em,wt,Rt,agl,bmi]])
    r=int(pred)
    if r>130:
        context={'value':r,'data':'Your Vehicle CO2 Emissions(g/km) is more than 130 g/km. \nso better to change vehicle\nTo make Air Quality.'}
        return render(request,'AdminApp/PredictedData.html',context)
    else:
        context={'value':r,'data':'Your Vehicle CO2 Emissions(g/km) is less than 130 g/km. \nIt is good vehicle.'}
        return render(request,'AdminApp/PredictedData.html',context)
```

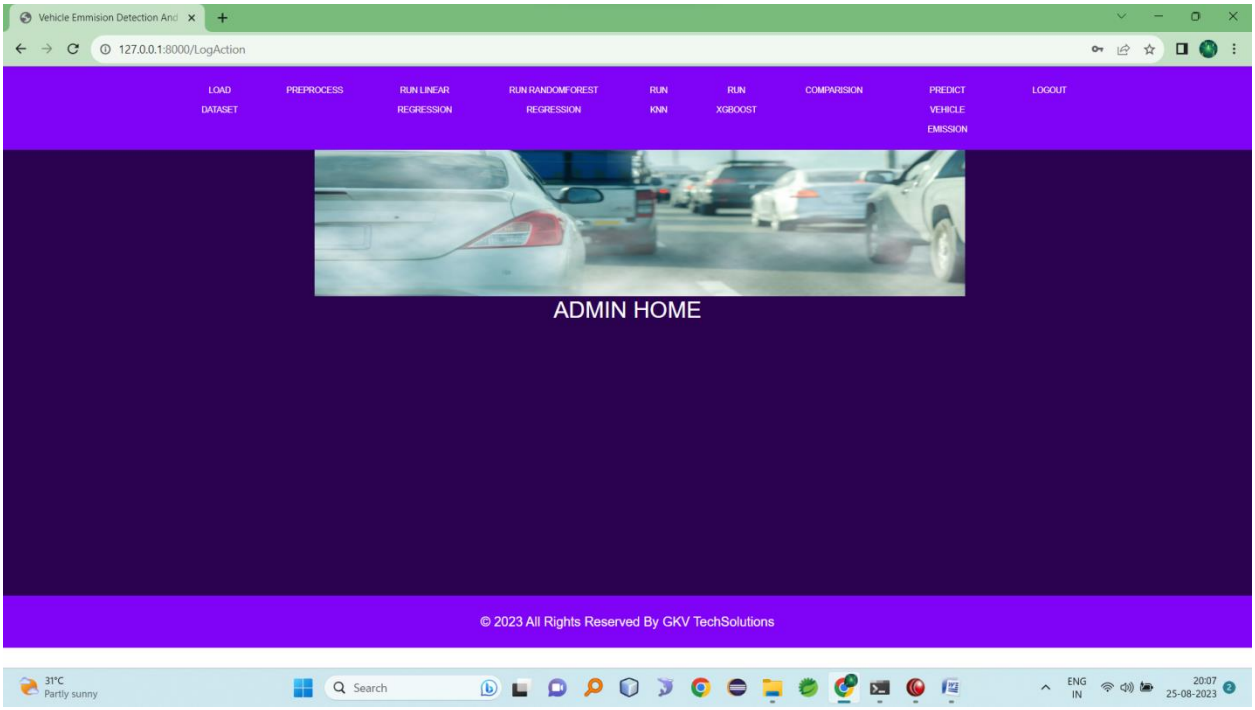
6. OUTPUT SCREENS



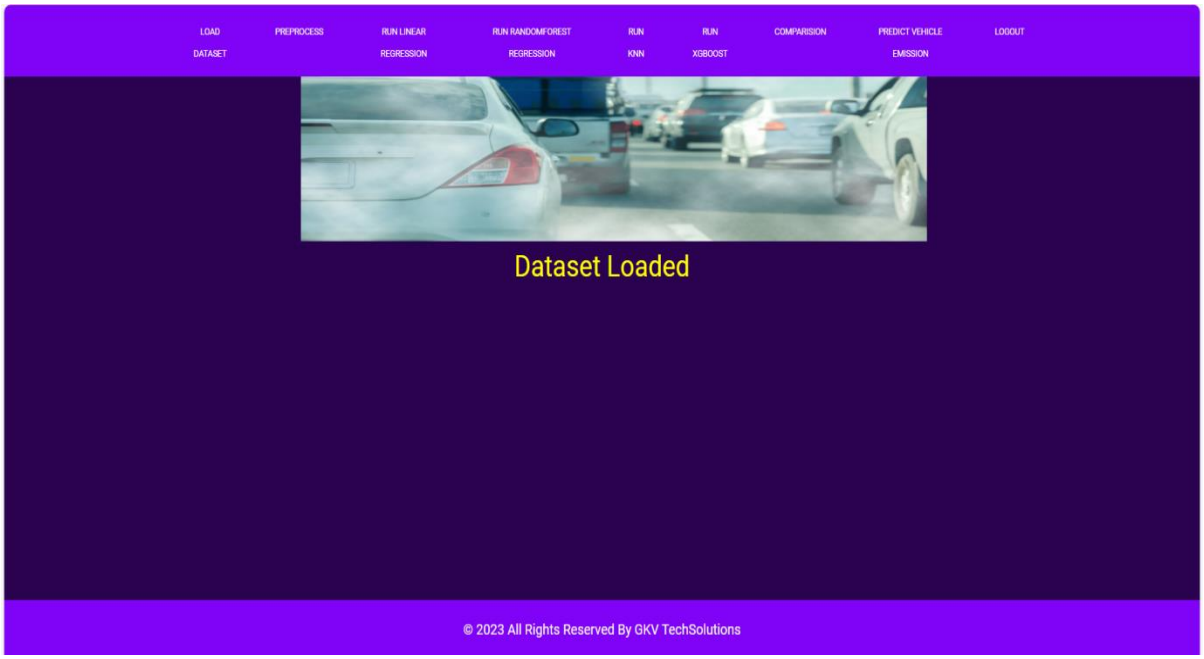
6.1.1.1 Index page



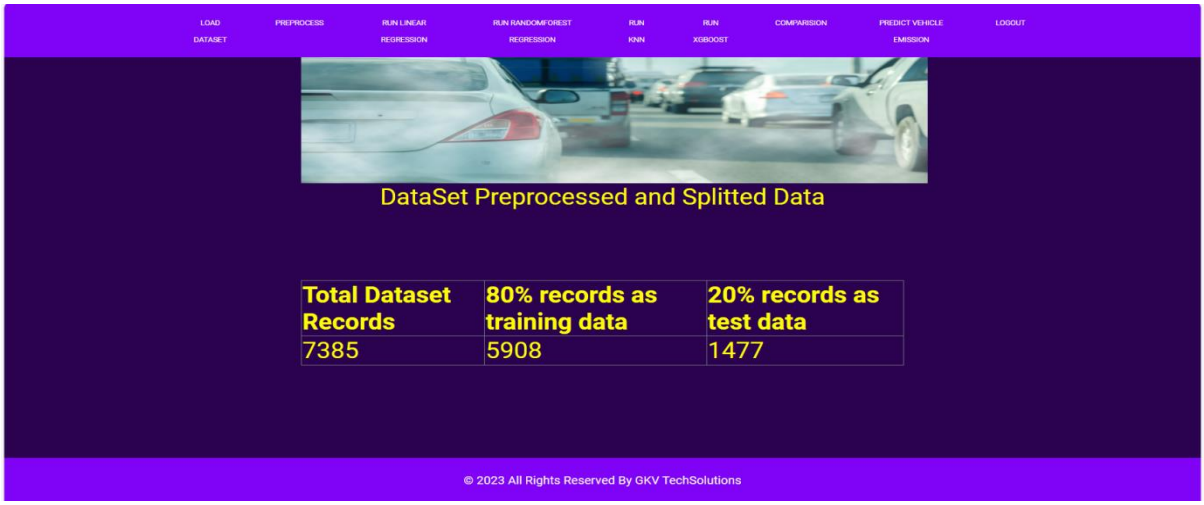
6.1.1.2 Admin Login



6.1.1.3 Home Page

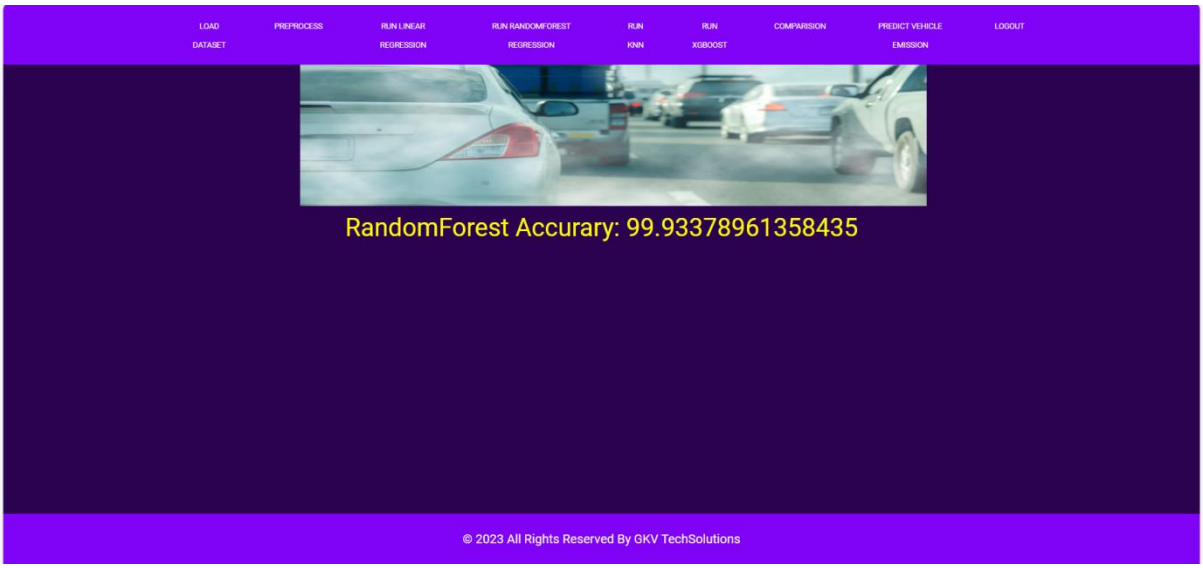


6.1.1.4 Dataset Loaded

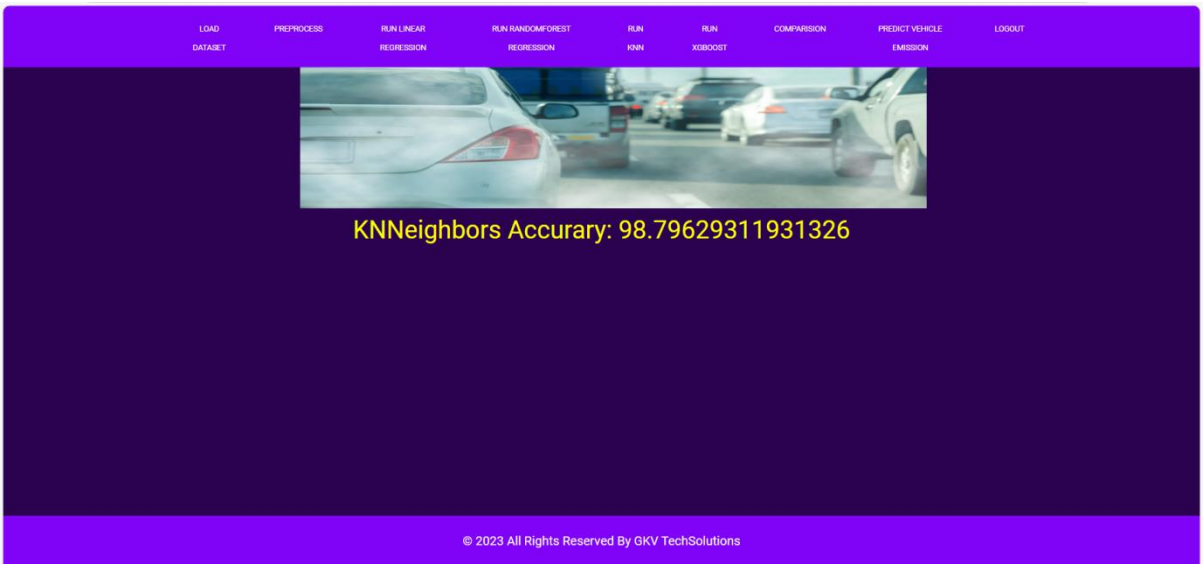


6.1.1.5 Pre-process Completes

6.1.1.6 Linear Regression

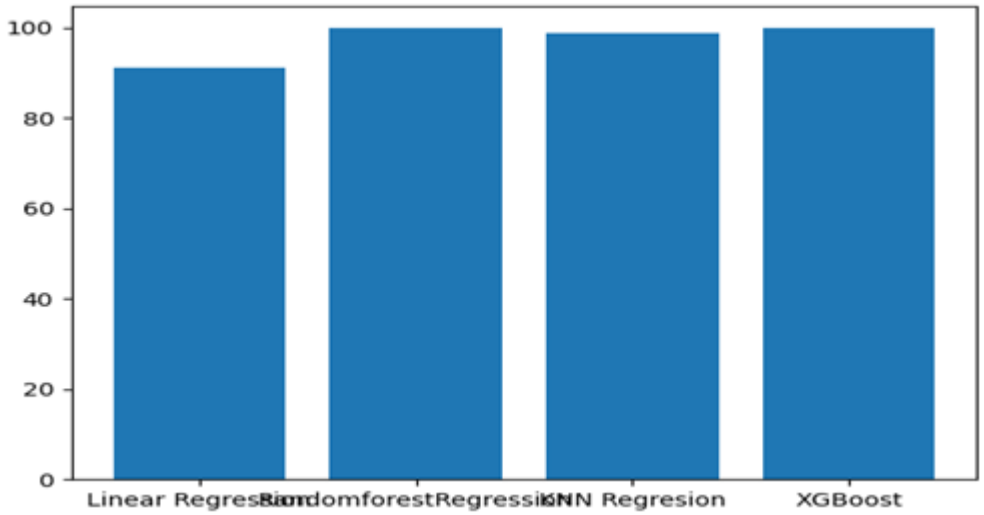


6.1.1.7 Random Forest Regression

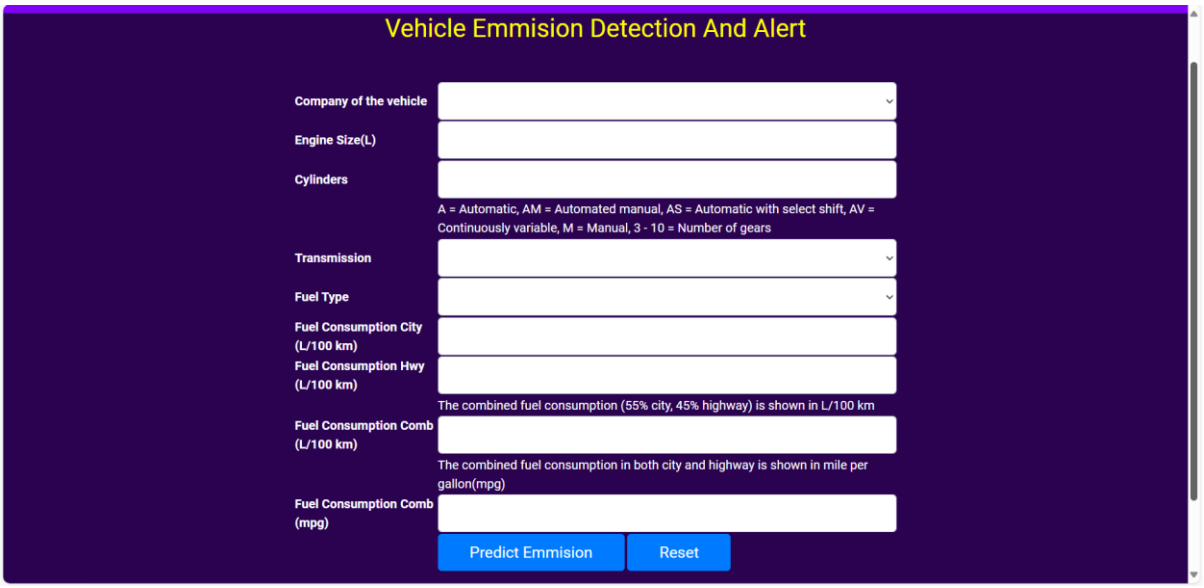


6.1.1.8 KNN neighbour Accuracy

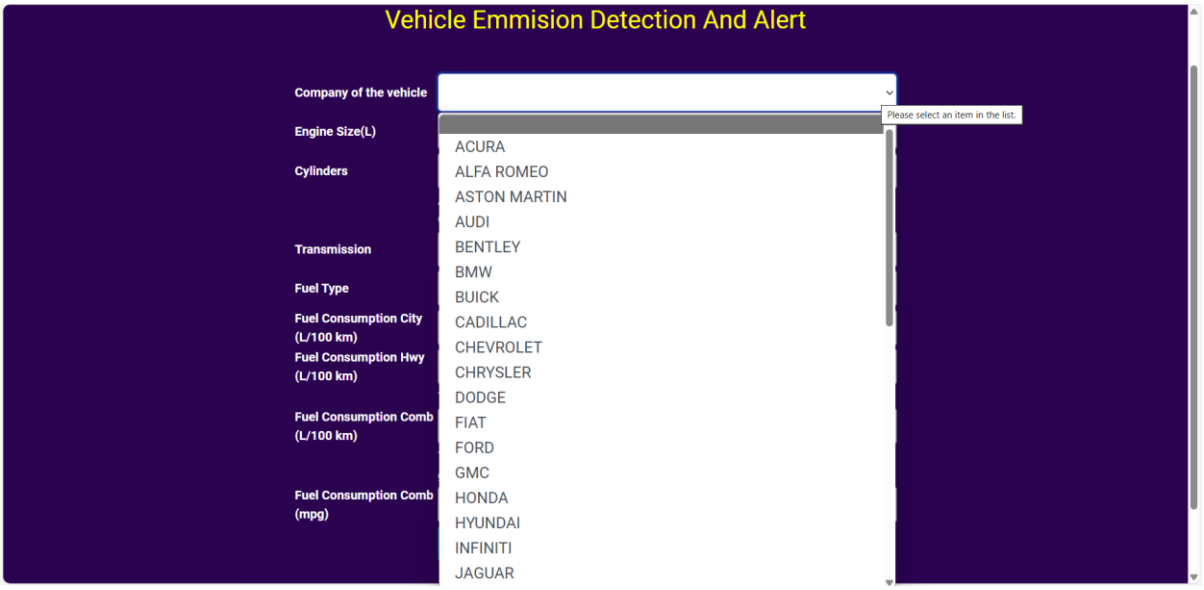
6.1.1.9 Run xgboost



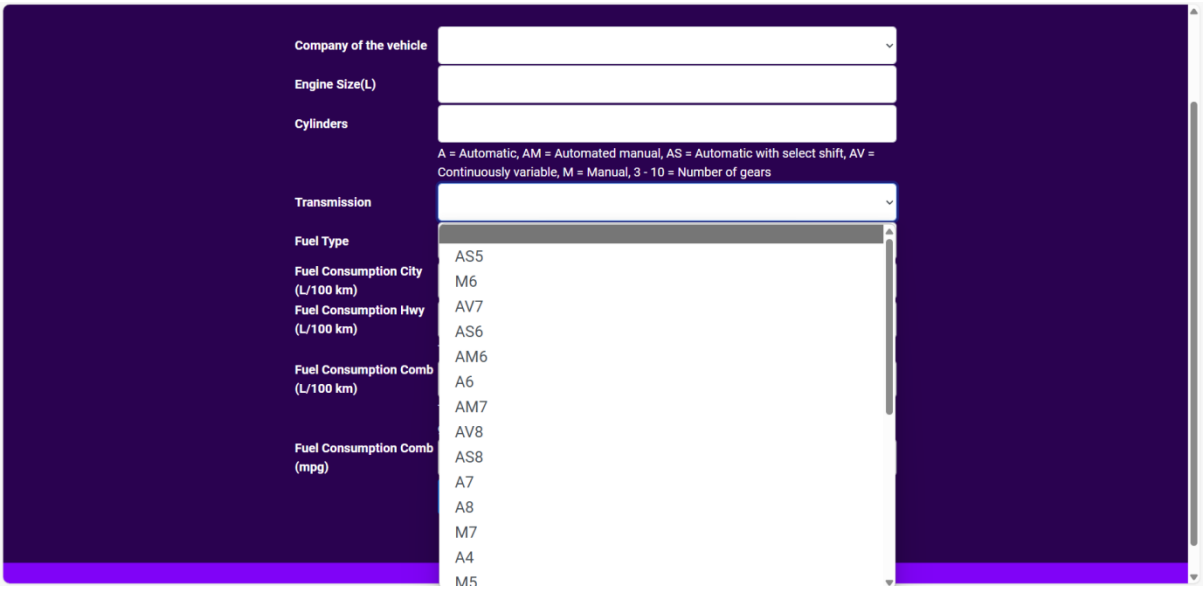
6.1.1.10 Comparison Graph



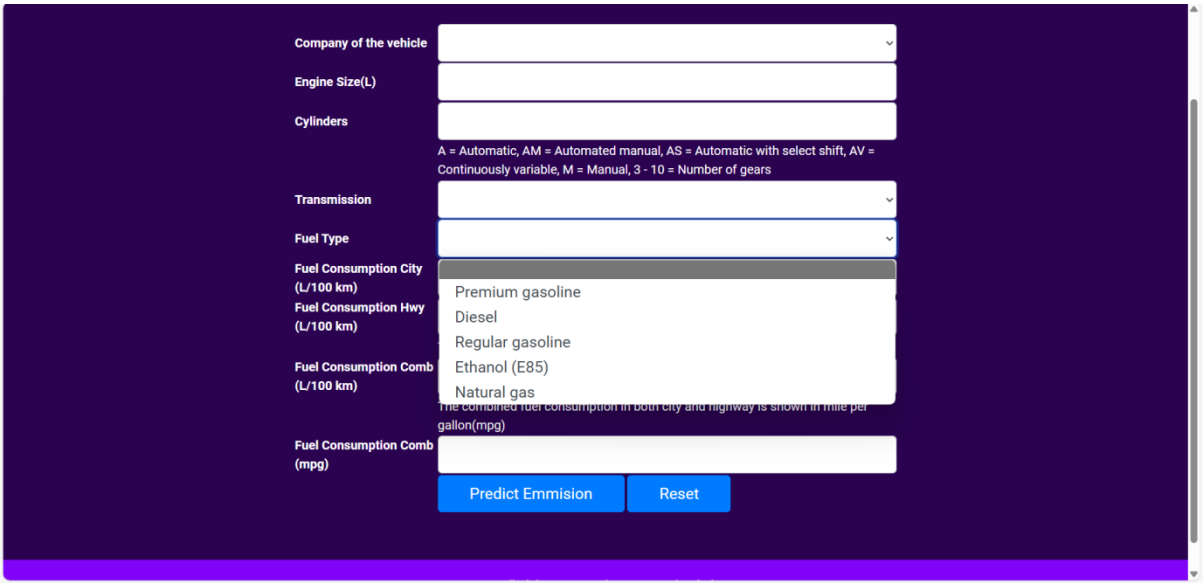
6.1.1.11Vehicle Details



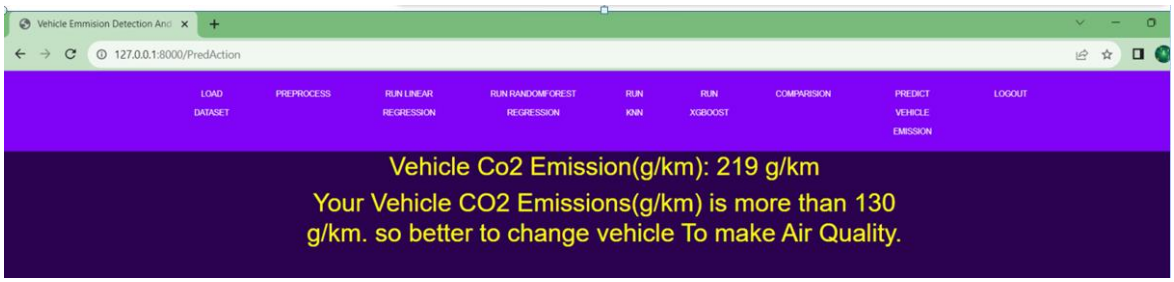
6.1.1.12 Vehicle Company



6.1.1.13 Vehicle Transmission



6.1.1.14 Vehicle Fuel Type



6.1.1.15 Output

7. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Types of Tests

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other

kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8. CONCLUSION

Environmental protection has been a hot topic in academic and industrial communities. This paper focuses on predicting the missing basic information of vehicles from telemetry data to monitor the vehicle emission. A variety of data mining methods are adopted to perform predictions based on the vehicle telemetry data provided by an environmental protection agency in a certain city and successfully made precise inferences on fuel type and gasoline-powered vehicle registration time. In the prediction for the registration time of diesel vehicles, the prediction accuracy rate just reaches about 70% due to the fact that the division of registration time is artificially controlled and the status of different vehicles varies a lot for different users. Further work will be carried out on the basis of more related data and improved algorithms to make more precise prediction on the vehicle emission-related information.

8.1 Future Enhancement

In the future, the vehicle emission detection system can be further enhanced by integrating IoT sensors directly into vehicles for real-time monitoring and reporting of emission levels. The system can be connected to government transport databases to automatically track non-compliant vehicles and issue fines or warnings. Additionally, mobile apps can be developed for vehicle owners to receive alerts, view emission reports, and get maintenance tips. Advanced machine learning models, such as deep learning, can also be introduced to improve prediction accuracy. Furthermore, implementing cloud-based storage and processing can make the system scalable and accessible from multiple locations for centralized monitoring.

9. REFERENCES

[1] S. A. Yashnik, S. P. Denisov, N. M. Danchenko, and Z. R. Ismagilov, "Synergetic effect of Pd addition on catalytic behavior of monolithic platinum-manganese-alumina catalysts for diesel vehicle emission control," *Applied Catalysis B: Environmental*, vol. 185, no. 15, pp. 322–336, 2016.

[2] E. M. Fujita, D. E. Campbell, B. Zielinska et al., "Comparison of the MOVES2010a, MOBILE6.2, and EMFAC2007 mobile source emission models with on-road traffic tunnel and remote sensing measurements," *Journal of the Air & Waste Management Association*, vol. 62, no. 10, pp. 1134–1149, 2012.

[3] J. S. Fu, X. Dong, Y. Gao, D. C. Wong, and Y. F. Lam, "Sensitivity and linearity analysis of ozone in East Asia: the effects of domestic emission and intercontinental transport," *Journal of the Air & Waste Management Association*, vol. 62, no. 9, pp. 1102–1114, 2012.

[4] A. Kfoury, F. Ledoux, C. Roche, G. Delmaire, G. Roussel, and D. Courcot, "PM2.5 source apportionment in a French urban coastal site under steelworks emission influences using constrained non-negative matrix factorization receptor model," *Journal of Environmental Sciences*, vol. 40, pp. 114–128, 2016.

[5] Y. Cheng, *Comparative Study on Sino-US Control Systems of Motor Vehicle Exhaust Pollution*, China University of Geosciences, Beijing, China, 2011.

[6] X. Wu, *Detection and Standard Revision of Exhaust Pollutants by Acceleration Simulation Mode for In-Use Gasoline Vehicle*, Chang'an University, Xi'an, China, 2012.

[7] G. A. Bishop, J. R. Starkey, A. Ihlenfeldt, W. J. Williams, and D. H. Stedman, "IR long-path photometry: a remote sensing tool for automobile emissions," *Analytical Chemistry*, vol. 61, no. 10, pp. 671A–677A, 1989.

[8] R. D. Stephens and S. H. Cadle, "Remote sensing measurements of carbon monoxide emissions from on-road vehicles," *Journal of the Air & Waste Management Association*, vol. 41, no. 1, pp. 39–46, 1991.