CO2 EMISSION RATING BY VEHICLES USING DATA SCIENCE TECHNIQUE

A PROJECT REPORT

Submitted by

DHANUSHPRIYA T [REGISTER NO:211419104059]

JECINTHA T [REGISTER NO:211419104112]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"CO2 emission rating by vehicles using data science technique"** is the bonafide work of "**DHANUSHPRIYA T(211419104059) JECINTHA T(211419104112)"**who carried out the project work under **Mrs.S.T.SANTHANALAKSMI** supervision.

SIGNATURE

Dr.L.JABASHEELA,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

S.T.SANTHANALAKSHMI,M.Tech,
SUPERVISOR
ASSOCIATE PROFESSOR
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We **DHANUSHPRIYA T(211419104031),JECINTHA T(211419104112)** hereby declare that this project report titled "**CO2 emission rating by vehicles using data science technique**", under the guidance of **Mrs.S.T.SANTHANALAKSHMI M.Tech.,** is the original work done by us and we have not plagiarized orsubmitted to any other degree in any university by us.

DHANUSHPRIYA T

JECINTHA T

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYA RAJESWARI**, **Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank our guide Mrs. **S.T.SANTHANALAKSHMI M.Tech.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

DHANUSHPRIYA T

JECINTHA T

ABSTRACT

Our personal vehicles are a major cause of global warming. Collectively, cars account for nearly one-fifth of all emissions, emitting around 24 pounds of carbon dioxide and other global-warming gases for every gallon of gas. About five pounds comes from the extraction, production, and delivery of the fuel, while the great bulk of heat-trapping emissions-more than 19 pounds per gallon-comes right out of a car's tailpipe A typical passenger vehicle emits about 4.6 metric tons of carbon dioxide per year. This number can vary based on a vehicle's fuel, fuel economy, and the number of miles driven per year. The higher the number of the controlled and uncontrolled effect variables that influence the co2 properties, the lesser the predicted accuracy. Despite this, a few experimental designs have been suggested by considering the controllable effect variables and interaction terms between them. To predict the emission of gas from cars we Supervised machine learning technique which is one of the great technique for predicting the CO2 emission rating. We have used Random Forest algorithm to get a best accuracy rate. In our model, the user will sign up into the user page to get to know about the CO2 emitted in the vehicle that the user has been using. The user will enter the relevant details and the output will be predicted. The predicted out will be displayed in the CO2 control inspector login page, where the inspector give feedback to the user in order to reduce the emission rate of the CO2 gas. The user interface provides a better understanding to our model.

LIST OF ABBREVIATIONS

CO2	-	Carbon Dioxide
RTA	-	Road and Transport Authority
IPCC	-	Intergovernmental Panel on Climate Change
CH4	-	Methane
N20	-	Nitrous oxide
HFC	-	Hydrofluorocarbons
PFC	-	Perfluorocarbons
SVR	-	Support Vector Regression
RMSE	-	Root Mean Square Error
GUI	-	Graphical User Interface
ERD	-	Entity Relationship Diagram
MVC	-	Model View Controller
XG Boost	-	eXtreme Gradient Boosting
FP	-	False Positives
FN	-	False Negatives
TP	-	True Positives
TN	-	True Negatives

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

The global warming issues now become a universal problem for all nations. The Intergovernmental Panel on Climate Change (IPCC) reported that scientists were more than 95% certain that most of global warming is caused by increasing concentrations of greenhouse gasses and other human (anthropogenic) activities [1]. That balance between earth and atmosphere affected by an increase in acid gasses charcoal or carbon dioxide (CO2), methane (CH4), nitrous oxide (N2O), hydrofluorocarbons (HFC) and perfluorocarbons (PFC) more commonly known by greenhouse gasses. In particular, CO2 is a major cause of Global Warming [2]. About eight billion tons per year of carbon in the form of CO2 emitted globally through burning fossil fuels for transport and for the production of heat and electricity around the world [3]. Emission of carbon dioxide is the rest results of combustion of water (H2O) and carbon monoxide gas (CO) or also called as carbon dioxide (CO2) that is a greenhouse gas.

Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammer bacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming

skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation.

This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

Fig 1.1

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is y = f(X). The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

1.2 PROBLEM DEFINITION

The data was collected through online resources to establish the project. such as some independent features. After collecting datasets from various resources, the dataset must be preprocessed before training to the model. The data

preprocessing can be done in various stages, beginning with reading the collected dataset the process continues to data cleaning. In data cleaning the datasets contain some redundant attributes, those attributes are not considered for CO2 emission prediction. So, we have to drop unwanted attributes and datasets containing some missing values. We need to drop these missing values or fill them with unwanted nan values in order to get better accuracy. Statistical algorithms and machine learning techniques to identify future outcomes based on historical data. The goal is to go beyond knowing what has happened to provide the best assessment of what will happen in the future. In our system, we used a supervised machine learning algorithm having subcategories as classification and regression. The classification algorithm will be most suitable for this system.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2
2. LITERATURE SURVEY

[1] A microscopic model of vehicle CO2 emissions based on deep learning – A spatiotemporal analysis of taxicabs in Wuhan, China.

It is important to assess environmental impact of intelligent transportation systems, and hence developing a vehicle emission model with high accuracy has been a long-standing topic in transportation research. However, current vehicle emission models are either overly simple using average speed, resulting in low estimation accuracy, or they are too complicated requiring excessive inputs, relying on too much prior knowledge. In this study, we develop and evaluate a deep learning-based vehicle emission model (DL-VEM) to estimate the instantaneous CO2 emissions of taxicabs. First, we examine the correlation between observed emissions and vehicle driving condition data collected in a PEMS experiment. Then, an end-to-end deep learning structure is developed to model patterns of vehicle emissions. Specifically, LSTM networks are used to learn temporal dependencies of historical driving patterns, and fully connected networks are employed to extract deep features of current driving behaviors and external environment. Our model aggregates the outputs of these networks using different learnable weights. Experiments were conducted in Wuhan, China, where our model was trained and validated using observed datasets. Compared with the state-of-the-art models, our model achieved higher accuracy in estimating CO2 emissions. Thereafter, it was applied to a taxicab trajectory dataset in one day, and spatiotemporal patterns of CO2 emissions were presented using different fuel types. Importantly, we find that an increment of 24.94% emissions can be expected if petrol instead of compressed natural gas was used by each taxicab in Wuhan.

[2] Analysis and Prediction Model of Fuel Consumption and Carbon Dioxide Emissions of Light-Duty Vehicles

Due to the alarming rate of climate change, fuel consumption and emission estimates are critical in determining the effects of materials and stringent emission control strategies. In this research, an analytical and predictive study has been conducted using the Government of Canada dataset, containing 4973 light-duty vehicles observed from 2017 to 2021, delivering a comparative view of different brands and vehicle models by their fuel

consumption and carbon dioxide emissions. Based on the findings of the statistical data analysis, this study makes evidence-based recommendations to both vehicle users and producers to reduce their environmental impacts. Additionally, Convolutional Neural Networks (CNN) and various regression models have been built to estimate fuel consumption and carbon dioxide emissions for future vehicle designs. This study reveals that the Univariate Polynomial Regression model is the best model for predictions from one vehicle feature input, with up to 98.6% accuracy. Multiple Linear Regression and Multivariate Polynomial Regression are good models for predictions from multiple vehicle feature inputs, with approximately 75% accuracy. Convolutional Neural Network is also a promising method for prediction because of its stable and high accuracy of around 70%. The results contribute to the quantifying process of energy cost and air pollution caused by transportation, followed by proposing relevant recommendations for both vehicle users and producers. Future research should aim towards developing higher performance models and larger datasets for building APIs and applications.

[3] Predicting CO2 Emission Using Machine Learning

The Random Forest & SVM are put forward to estimate the outlay of CO2 outpouring. Energy expenditure, such as power and coal energy, is the cause of the aggressive growth in CO2 emissions. The aim is to track co2 emissions gleaned from electrical energy and coal utilized in the manufacturing process. In setup to train and test the model, the statistics on electrical and energy were obtained. The statistics were separated into 60%training data and 40% testing data. To find the distinguished parameters of both model's, trial and error approach was used. The minimal error of the model reflects a more precise estimation while using RMSE to measure the model's error.

[4] The Prediction of Carbon Emission Information in Yangtze River Economic Zone by Deep Learning

This study aimed to respond to the national "carbon peak" mid-and long-term policy plan, comprehensively promote energy conservation and emission reduction, and accurately manage and predict carbon emissions. Firstly, the proposed method analyzes the Yangtze River Economic Belt as well as its "carbon peak" and carbon emissions. Secondly, a support vector regression (SVR) machine prediction model is proposed for the

carbon emission information prediction of the Yangtze River Economic Zone. This experiment uses a long short-term memory neural network (LSTM) to train the model and realize the experiment's prediction of carbon emissions. Finally, this study obtained the fitting results of the prediction model and the training model, as well as the prediction results of the prediction model. Information indicators such as the scale of industry investment, labor efficiency output, and carbon emission intensity that affect carbon emissions in the "Yangtze River Economic Belt" basin can be used to accurately predict the carbon emissions information under this model. Therefore, the experiment shows that the SVR model for solving complex nonlinear problems can achieve a relatively excellent prediction effect under the training of LSTM. The deep learning model adopted herein realized the accurate prediction of carbon emission information in the Yangtze River Economic Zone and expanded the application space of deep learning. It provides a reference for the model in related fields of carbon emission information prediction, which has certain reference significance.

[5] Carbon dioxide emission prediction using support vector machine

In this paper, the SVM model was proposed for predict expenditure of carbon (CO2) emission. The energy consumption such as electrical energy and burning coal is input variable that affect directly increasing of CO2 emissions were conducted to built the model. Our objective is to monitor the CO2 emission based on the electrical energy and burning coal used from the production process. The data electrical energy and burning coal used were obtained from Alcohol Industry in order to training and testing the models. It divided by cross-validation technique into 90% of training data and 10% of testing data. To find the optimal parameters of SVM model was used the trial and error approach on the experiment by adjusting C parameters and Epsilon. The result shows that the SVM model has an optimal parameter on C parameters 0.1 and 0 Epsilon. To measure the error of the model by using Root Mean Square Error (RMSE) with error value as 0.004. The smallest error of the model represents more accurately prediction. As a practice, this paper was contributing for an executive manager in making the effective decision for the business operation were monitoring expenditure of CO2 emission.

[6] A Novel GDP Prediction Technique based on Transfer Learning using CO2 Emission Dataset1

In the last 150 years, CO2 concentration in the atmosphere has increased from 280

parts per million to 400 parts per million. This has caused an increase in the average global temperatures by nearly 0.7 °C due to the greenhouse effect. However, the most prosperous states are the highest emitters of greenhouse gases (especially CO2). This indicates a strong relationship between gaseous emissions and the gross domestic product (GDP) of the states. Such a relationship is highly volatile and nonlinear due to its dependence on the technological advancements and constantly changing domestic and international regulatory policies and relations. To analyse such vastly nonlinear relationships, soft computing techniques has been quite effective as they can predict a compact solution for multi-variable parameters without any explicit insight into the internal system functionalities.This paper reports a novel transfer learning based approach for GDP prediction, which we have termed as 'Domain Adapted Transfer Learning for GDP Prediction. In the proposed approach per capita GDP of different nations is predicted using their CO2 emissions via a model trained on the data of any developed or developing economy. Results are comparatively presented considering three well-known regression methods such as Generalized Regression Neural Network, Extreme Learning Machine and Support Vector Regression. Then the proposed approach is used to reliably estimate the missing per capita GDP of some of the war-torn and isolated countries.

[7] Creating an emission model based on portable emission measurement system for the purpose of a roundabout.

Nowadays, climate change is believed to be the second most important problem faced by the global community (European Commission 2011). The main greenhouse gases emitted by human activities are carbon dioxide, methane, nitric oxide, hydrofluorocarbons, perfluorocarbons, and sulfur hexafluoride (Venkataraman et al. 2012; Wei et al. 2008; Lindley, McCulloch 2005).Transport is the main sector of the economy that causes environmental pollution and climate change. Emissions from transport, mainly road transport, make a significant contribution to the number of greenhouse gases in the atmosphere (OECD 2002). The automotive industry is the second largest sector producing CO_2 with a total share of 22% in general (IEA 2012).The World Health Organization estimates that around 2.4 million people lose their lives worldwide due to environmental pollution (WHO 2007).This paper presents an analysis of emission data from the PEMS system for real driving cycles of various types of vehicles, complying with EURO2-EURO6 standards, fueled with petrol, LPG, and diesel in urban, rural, and motorway areas as well as detailing roundabouts. The results show that in the range of roundabouts, there

is an increased emission of harmful exhaust components, such as CO_2, THC, CO, and NO*x*. Due to the specific traffic conditions that prevail at the roundabout (acceleration, braking, acceleration to a certain speed), the methodology for creating an exhaust emission model for this type of objects has been proposed. Statistical analysis of the received boosted regression tree models based on the coefficient of regression, root mean square error, and mean absolute error and based on the visual assessment of the results show that the obtained models are well represented by real data. The obtained results of emission calculations on roundabouts may be used to identify areas of increased emission of harmful exhaust components, as well as an introduction to prepare new roundabout design guidelines concerning emission data.

[8] Spatiotemporal patterns of carbon emissions and taxi travel using GPS data in Beijing.

With improved information and communication technologies, as well as location-based services (LBS) such as mobile phone communications, social software, vehicle-carried GPS (Global Position System) positioning terminals, etc., large-scale, high-quality, and consecutive spatiotemporal trajectory data on urban mobility has become an increasingly popular dataset and principal resource. Many researchers employ advanced data mining techniques and big geospatial data, among which taxi GPS data is one of the prevailing resources, to analyze individual travel patterns, the organization and planning of urban public spaces, construction of smart cities, and so forth. At present, studies using taxi GPS data include, but are not limited to, the following aspects: route planning and path-finding, traffic operational state identification, identification of origin-destination (OD) and the clustering method, and taxi fuel consumption and emissions estimation. This study uses taxi GPS data to reconstruct taxi trajectories in Beijing. We then use the carbon emission calculation model based on a taxi fuel consumption algorithm and the carbon dioxide emission factor to calculate emissions and apply a visualization method called kernel density analysis to obtain the dynamic spatiotemporal distribution of carbon emissions. Total carbon emissions show substantial temporal variations during the day, with maximum values from 10:00–11:00 (57.53 t), which is seven times the minimum value of 7.43 t (from 03:00–04:00). Carbon emissions per kilometer at the network level are steady throughout the day (0.2 kg/km). The Airport Expressway, Ring Roads, and large intersections within the 5th Ring Road maintain higher carbon emissions than other areas.

[9] Prediction of CO2 emissions using deep learning hybrid approach: A case study in Indian context .

Carbon Dioxide (CO_2), which accounts for 1% of the atmospheric gases but responsible for about 81% of the total emissions, is a major component of Green House Gas (GHG) Emissions. Produced naturally (decomposition, respiration and ocean drive) or released as a consequence of human activities (burning of fossil fuels, cement production, automotive exhausts etc.), CO_2 has played a critical role in global warming. The concern over CO_2 is the significant change in its level over a short period of time. The net result is melting of polar ice caps or rising annual global temperature. India, a developing nation and an emerging economy, is the fourth largest producer of Carbon Dioxide emissions following China, United States of America and the European Union. India has now overtaken Russia to become the third largest producer of electricity but it still relies on coal as the biggest source of electricity. Awaken by the adversities of Carbon Dioxide Emissions, India has signed the Paris Agreement and pledged to reduce the Carbon Dioxide levels to 30-35% of the level in the year 2005. This agreement will start from the year 2020. The research is aimed at predicting the CO_2 levels in the year 2020 to have a better understanding of the challenge posed to Government of India. The prediction technique used is a deep learning hybrid model of Convolution Neural Network and Long Short-Term Memory Network (CNN-LSTM).

[10] CO2 emissions forecasting in multi-source power generation systems using dynamic Bayesian network.

Climate change is one of the significant challenges that the planet is facing nowadays. CO 2 emission is the largest contributor, and it is mainly released by human activities. In Europe, the energy sector is responsible for roughly two-thirds of all greenhouse gas (GHG) emissions and the amount of CO 2 emitted from electricity production can greatly vary in time as a function of sources used to generate it. An accurate prediction of CO 2 emissions not only provides a basis for policymakers, but it can also assist the management of carbon emissions in making efforts towards limiting emissions generation and global warming as a consequence. For such a purpose, researchers have proposed the use of traditional algorithms for forecasting CO 2 emissions from the energy

sector. However, there still are challenges yet to be overcome as regards forecasting CO 2 emissions from the energy sector. Power dispatch problem consists in planning the use of all available sources for minimizing the environmental impact while at the same time satisfying the energy demand, stressing the necessity to dealing with this topic in multi-source power generation systems. In this context, this paper presents the use of discrete Dynamic Bayesian Networks (DBN) to forecast CO 2 emissions in a multi-source power generation system. The proposed methodology has been evaluated using the multi-source Germany grid data. The results were benchmarked against Multilayer Perceptron (MLP), K-nearest neighbor algorithm (KNN) and Random Forest (RF), and it was found that DBN achieved a significantly better performance due to reducing average NRMSE by 16.57%, average MAE by 19.88 gCO2eq/kWh and average MedAE by 27.48 gCO2eq/kWh in comparison with the second best method.

CHAPTER 3
SYSTEM ANALYSIS

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

It is important to assess the environmental impact of intelligent transportation systems, and hence developing a vehicle emission model with high accuracy has been a long-standing topic in transportation research. However, current vehicle emission models are either overly simple using average speed, resulting in low estimation accuracy, or they are too complicated requiring excessive inputs and relying on too much prior knowledge. This study proposed a microscopic emission model based on deep learning for estimating instantaneous vehicle CO2 emissions with high accuracy. We first used the technique of dynamic time wrapping to time-align the sequences of the observed driving condition data and the measured emission data, owing to their out-of-synchronization. Then, according to the correlation of vehicle CO2 emissions and driving condition data, we developed our network structure with three parts including the historical, current, and external parts.

Disadvantages:

- This process is very complicated.
- The analysis process is only done.
- They find the CO2 emission rate via GPS, It's not a proper way to find the CO2 emission rate.
- The accuracy rate is very low.

3.2 PROPOSED SYSTEM

The data was collected through online resources to establish the project. such as some independent features. After collecting datasets from various resources, the dataset must be preprocessed before training to the model. The data preprocessing can be done in various stages, beginning with reading the collected dataset the process continues to data cleaning. In data cleaning the datasets contain some redundant attributes, those attributes are not considered for CO2 emission prediction. So, we have to drop unwanted attributes and datasets containing some missing values. We need to drop these missing values or fill them with unwanted nan values in order to get better accuracy. Statistical algorithms and machine learning techniques to identify future outcomes based on historical data. The goal is to go beyond knowing what has happened to provide the best assessment of what will happen in the future. In our system, we used a supervised machine learning algorithm having subcategories as classification and regression. The classification algorithm will be most suitable for this system.

Advantages:

- We are using the machine learning technique for getting better predictions.
- More than two machine learning algorithms are compared.
- We calculate the performance metrics.

3.3 FEASIBILITY STUDY

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

This dataset contains 950 records of features extracted from Vehicles, which were then used to find the smog rating from the vehicles.

Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

Building the classification model

The prediction of CO2 emission rating, a high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical, and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

Construction of a Predictive Model

Machine learning needs data gathering have lot of past data's. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to pre-process then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with

improving the accuracy.

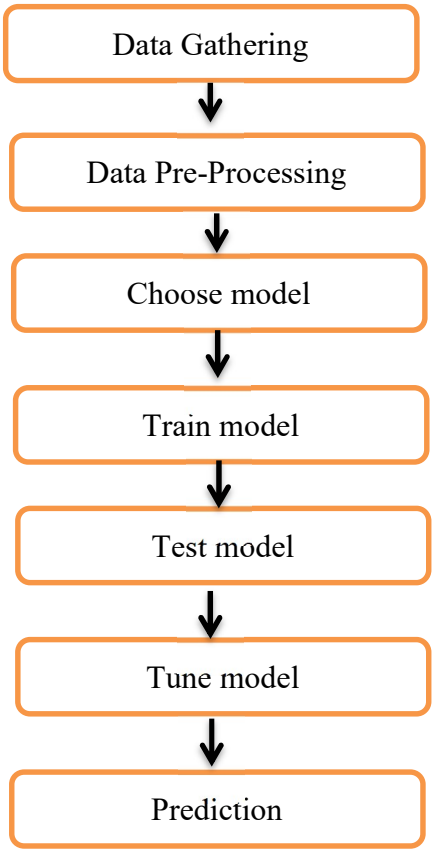


Fig 3.1 Process diagram

3.4 HARDWARE ENVIRONMENT

Project Requirements

General:

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements

A. Hardware requirements

B. software requirements

Functional requirements:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

Non-Functional Requirements:

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

HARDWARE ENVIRONMENT

- Processor : Intel i3 or later
- Hard disk : minimum 10 GB
- RAM : minimum 4 GB

3.5 SOFTWARE ENVIRONMENT

- Operating System : Windows 10 or later
- Tool :Anaconda with Jupyter Notebook

SOFTWARE DESCRIPTION

Anaconda is a free and distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes

more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook

- VSCode
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Conda :

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. These softwares have great graphical user interface and these will make your work easy to do. You can also use it to run your python script. These are the software carried by anaconda navigator.

JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which

contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.
Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called Anaconda
Running the Jupyter Notebook
Launching *Jupyter Notebook App:* The Jupyter Notebook App can be launched by clicking on the *Jupyter Notebook* icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): "jupyter notebook"
This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

JUPYTER Notebook App:

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser.The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.

Kernel: A notebook *kernel* is a "computational engine" that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist .When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results.Depending on the type of computations,

the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

PYTHON

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

CHAPTER 4

SYSTEM DESIGN

CHAPTER 4
SYSTEM DESIGN

4.1 ER DIAGRAM

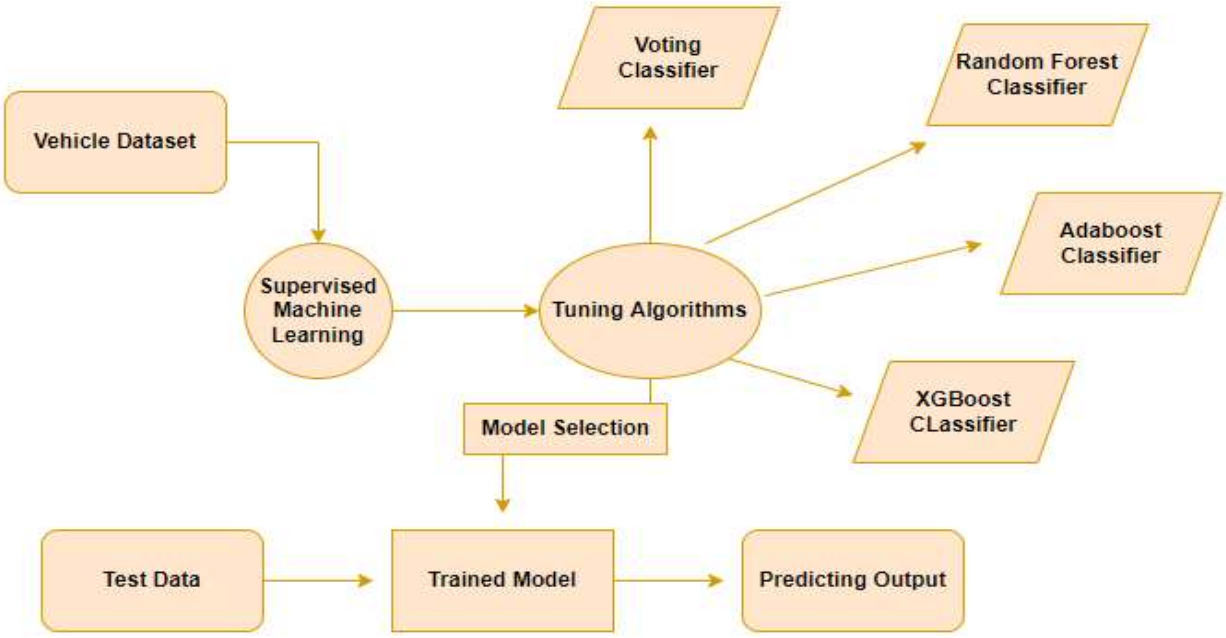


Fig 4.1 Entity Relationship diagram for CO2 emission rating by vehicles

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

4.2 DATA FLOW DIAGRAM

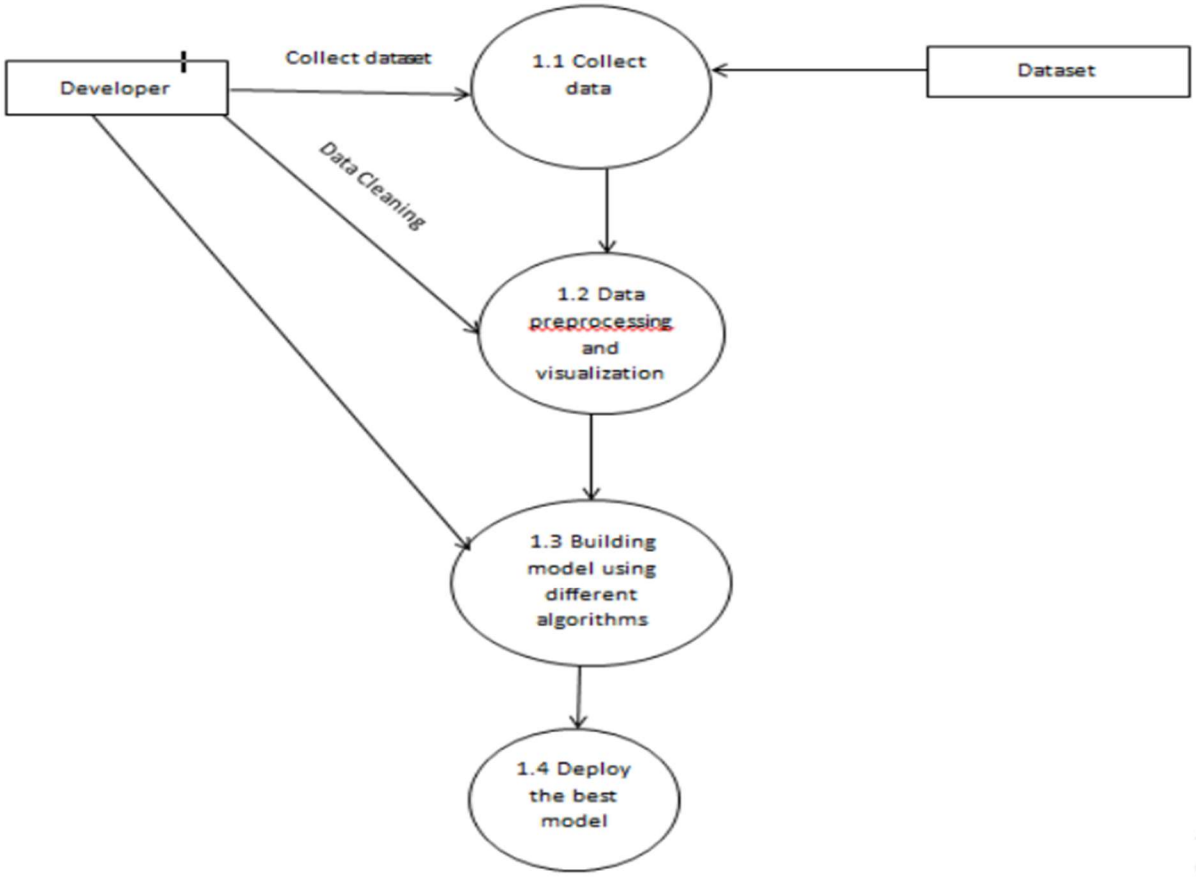


Fig 4.2 Dataflow diagram for CO2 emission rating by vehicles

4.3 UML DIAGRAMS

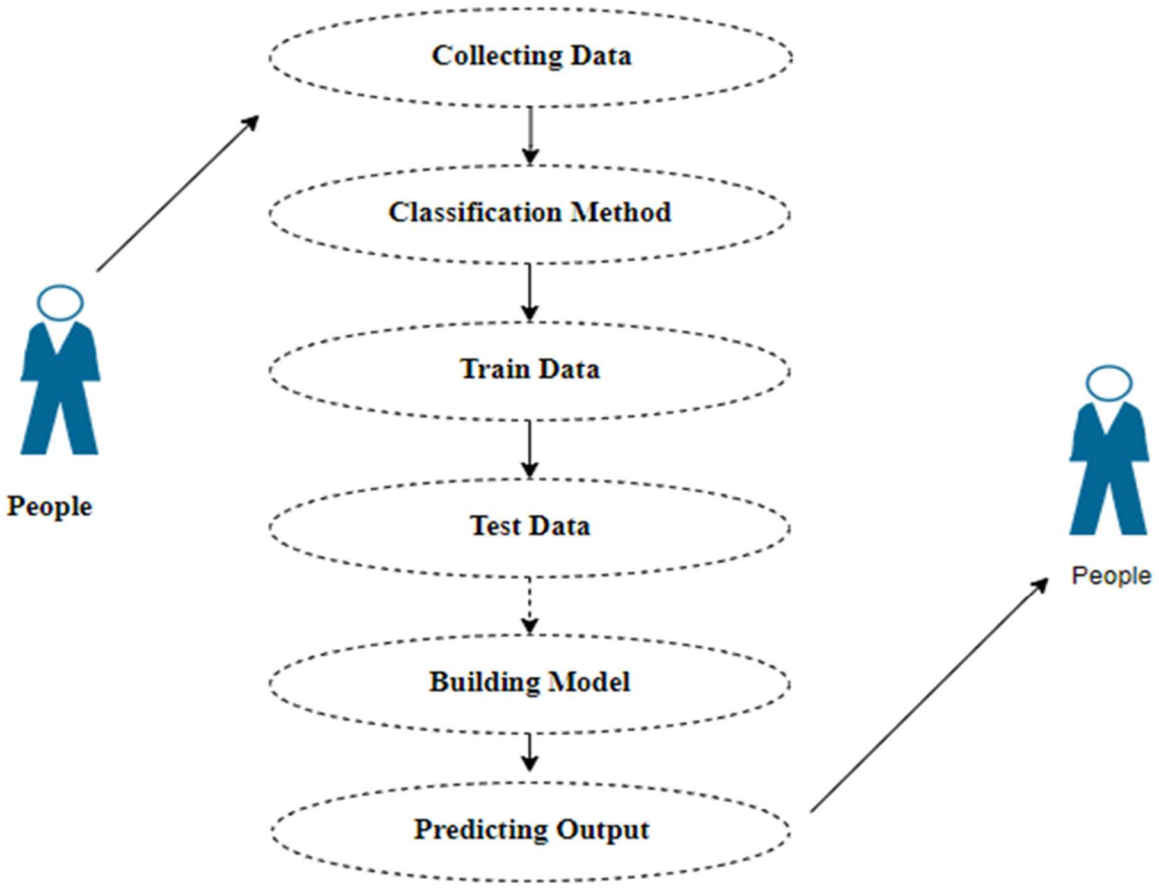


Fig 4.3 Use Case Diagram for CO2 emission rating by vehicles

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

4.4 SEQUENCE DIAGRAM

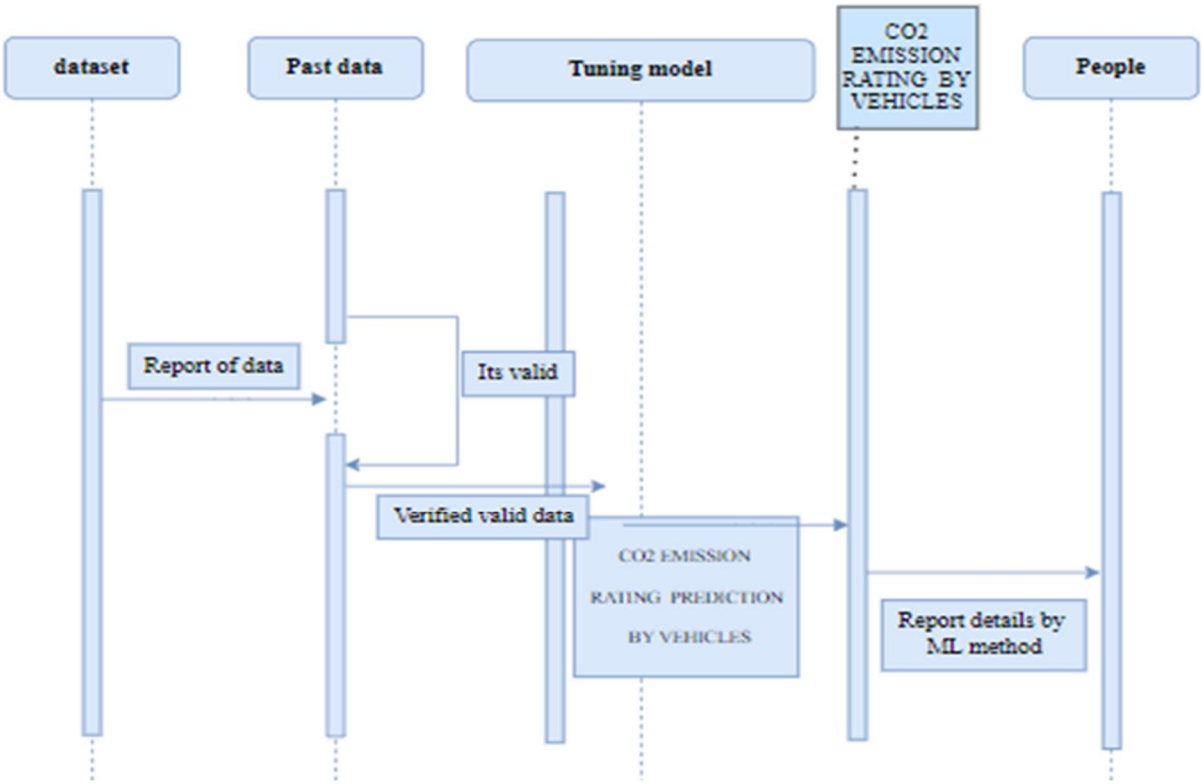


Fig 4.4 Sequence diagram for CO2 emission rating by vehicles

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

4.5 CLASS DIAGRAM

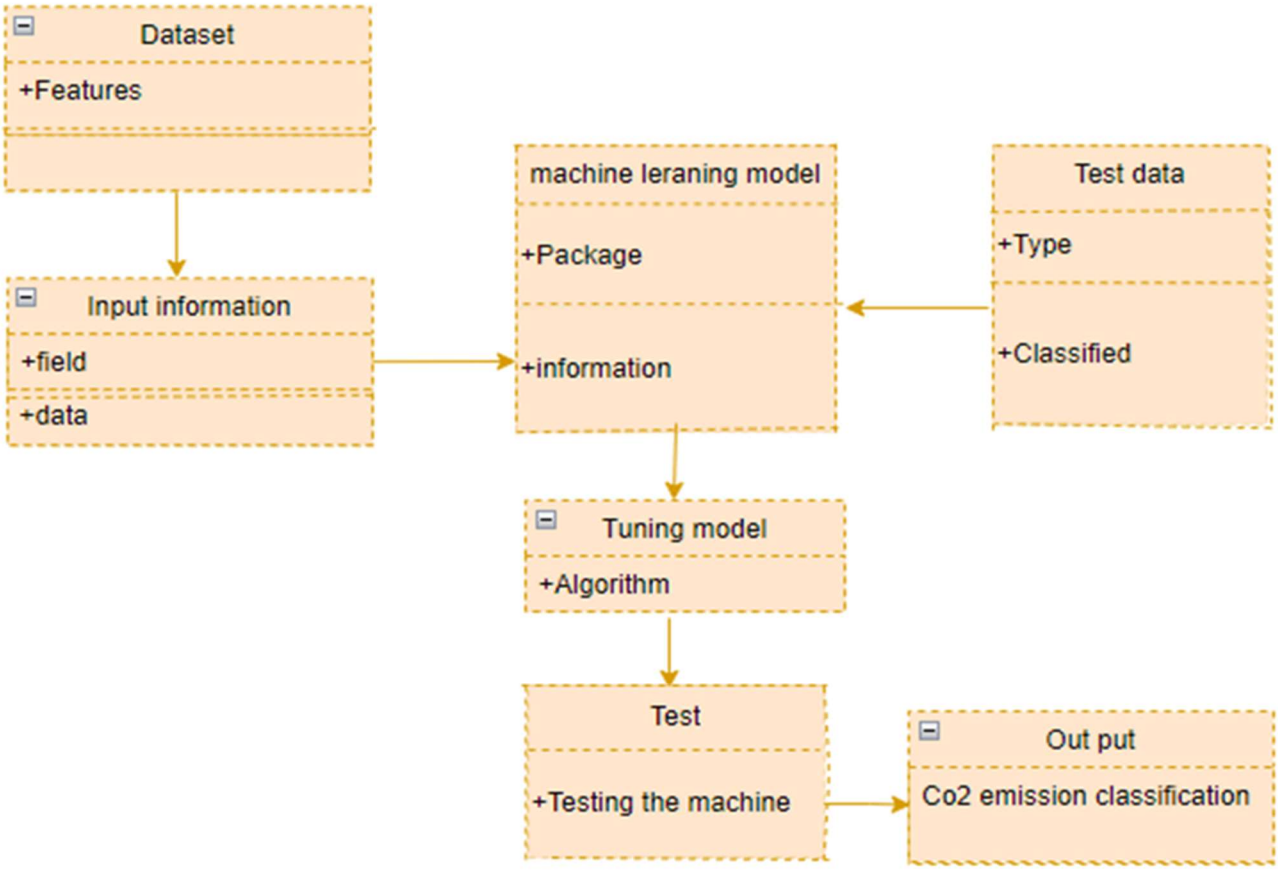


Fig 4.5 Class diagram for CO2 emission rating by vehicles

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder.

4.6 ACTIVITY DIAGRAM

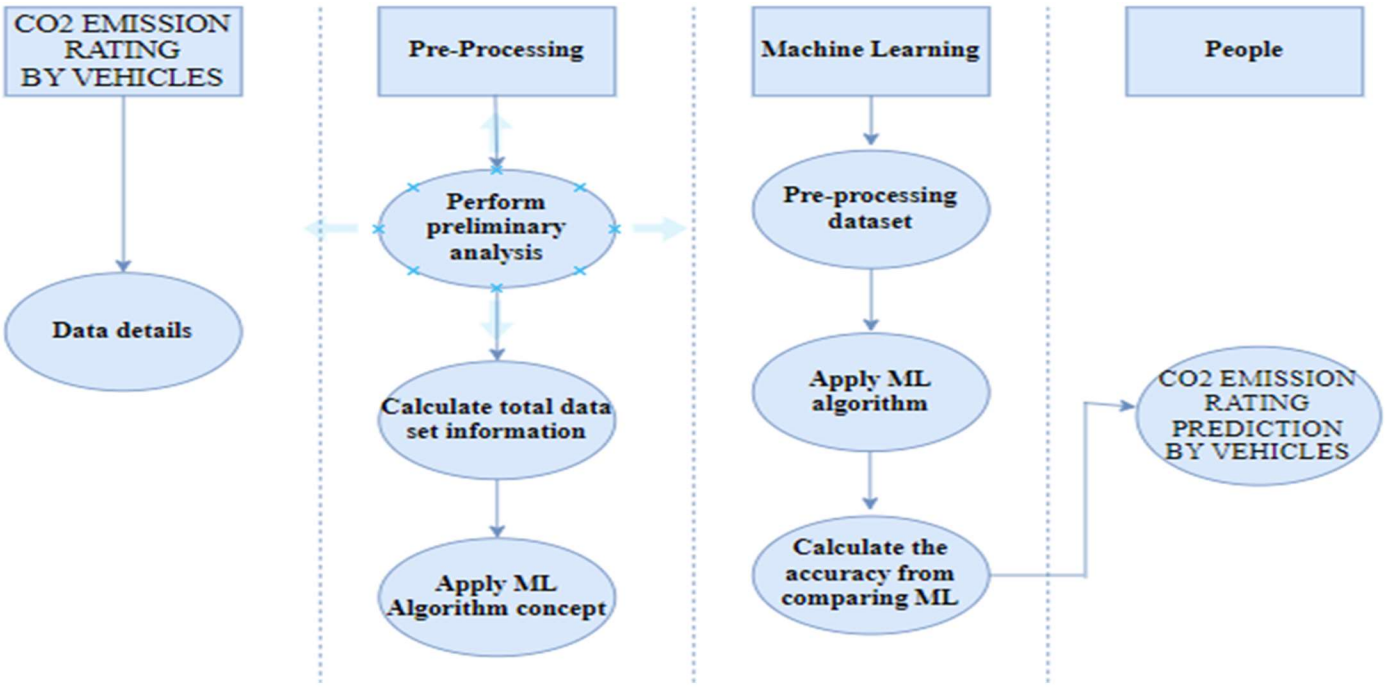


Fig 4.6 Activity diagram for CO2 emission rating by vehicles

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not.

CHAPTER 5

SYSTEM ARCHITECTURE

5.1 MODULE DESIGN SPECIFICATION

SYSTEM ARCHITECTURE

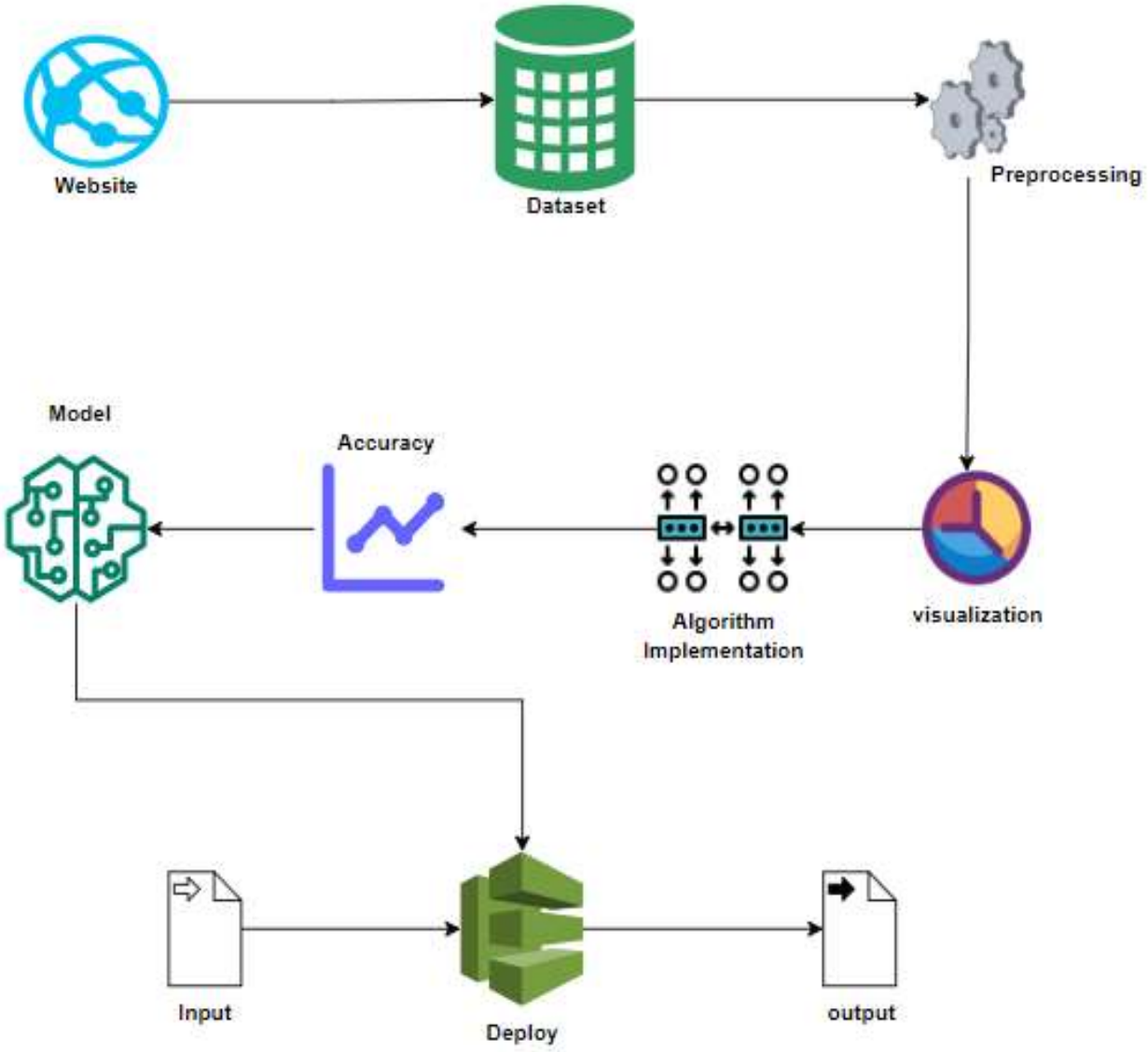


Fig 5.1 System architecture diagram for CO2 emission rating by vehicles

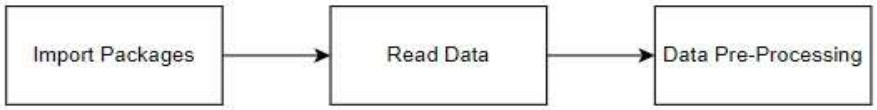
MODULE DESCRIPTION:

Data Pre-processing:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task**,** missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data.

MODULE DIAGRAM



Data visualization:

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

MODULE DIAGRAM

Fig 5.3 Data Visualization diagram

Deployment:

Deploying the model in Django Framework and predicting output. In this module the trained machine learning model is converted into pickle data format file (.pkl file) which is then deployed in our django framework for providing better user interface and predicting the output of how much the given data is emitting Co2.

MODULE DIAGRAM

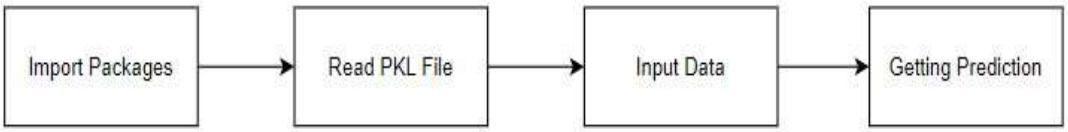


Fig 5.4 Deployment diagram

Flask (Web FrameWork):

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.However, Flask supports extensions that can add application features as if they were implemented in Flask itself.Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

FEATURES:

Flask was designed to be easy to use and extend. The idea behind Flask is to build a solid foundation for web applications of different complexity. From then on you are free to plug in any extensions you think you need. Also you are free to build your own modules. Flask is great for all kinds of projects. It's especially good for prototyping. Flask depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit. Plus Flask gives you so much more control on the development stage of project. It follows the principles of minimalism and let you decide how you will build your application.

- Flask has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
- ORM-agnostic: you can plug in your favourite ORM e.g. SQLAlchemy.
- Basic foundation API is nicely shaped and coherent.

The configuration is even more flexible than that of Flask, giving you plenty of solution for every production need.To sum up, Flask is one of the most polished and feature-rich micro frameworks, available. Still young, Flask has a thriving community, first-class extensions, and an elegant API**.** Flask comes with all the benefits of fast templates, strong WSGI features, thorough unit testability at the web application and library level, extensive documentation. So next time when starting a new project where you need some good features and a vast number of extensions, definitely check out Flask. Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Flask framework and is also easier to learn because it has less base code to implement a simple web-Application

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.Overview of Python Flask Framework Web apps are developed to generate content based on retrieved data that changes based on a user's interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Advantages of Flask:

- Higher compatibility with latest technologies.
- Technical experimentation.
- Easier to use for simple cases.
- Codebase size is relatively smaller.

Start Using an API

1. Most APIs require an API key. ...
2. The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.
3. The next best way to pull data from an API is by building a URL from existing API documentation.

Parameters

- **rule** (*str*) – The URL rule string.
- **endpoint** (*Optional[str]*) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to view_func.__name__.
- **view_func** (*Optional[Callable]*) – The view function to associate with the endpoint name.
- **provide_automatic_options** (*Optional[bool]*) – Add the OPTIONS method and respond to OPTIONS requests automatically.
- **options** (*Any*) – Extra options passed to the Rule object.

Return type -- None

After_Request(f)

Register a function to run after each request to this object.The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.If a function raises an exception, any remaining after request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use teardown_request() for that

Parameters:

f (*Callable[[Response], Response]*)

Return type

Callable[[Response], Response]

after_request_funcs: t.Dict[AppOrBlueprintKey,

t.List[AfterRequestCallable]]

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the after_request() decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

app_context()

Create an AppContext. Use as a with block to push the context, which will make current_app point at this application.An application context is automatically

pushed by RequestContext.push() when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

With app.app_context():

Init_db()

HTML

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 ALGORITHM:

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

The below 4 different algorithms are compared:

- AdaBoost
- XGBoost
- Random Forest
- Voting Classifier

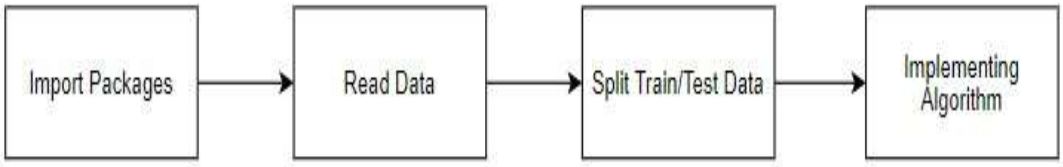
Adaboost Classifier:

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem. The most suited

and therefore most common algorithm used with AdaBoost are decision trees with one level. How does the AdaBoost algorithm work explain? It works on the principle of learners growing sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. The AdaBoost algorithm works on the same principle as boosting with a slight difference.

```
ab = AdaBoostClassifier()
ab.fit(X_train,y_train)
predicted_ab = ab.predict(X_test)
```

MODULE DIAGRAM

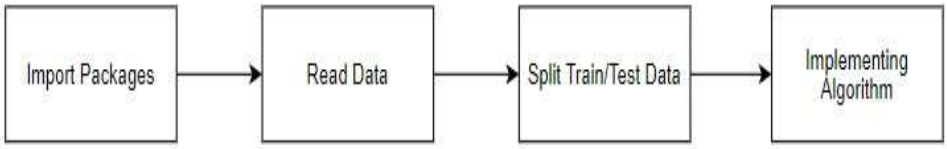


XG Boost:

XGBoost (eXtreme Gradient Boosting) is a popular supervised-learning algorithm used for regression and classification on large datasets. It uses sequentially-built shallow decision trees to provide accurate results and a highly-scalable training method that avoids overfitting. It is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems. xgboost will generally fit training data much better than linear regression, but that also means it is prone to overfitting, and it is less easily interpreted. Either one may end up being better, depending on your data and your needs.

```
xgb = XGBClassifier()
xgb.fit(X_train,y_train)
predicted_xgb = xgb.predict(X_test)
```

MODULE DIAGRAM



Random Forest Classifier

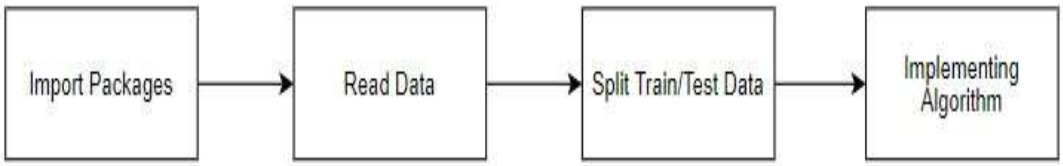
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees,* resulting in a *forest of trees,* hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.
The following are the basic steps involved in performing the random forest algorithm:

- Pick N random records from the dataset.
- Build a decision tree based on these N records.
- Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

```
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
predicted_rf = rf.predict(X_test)
```

MODULE DIAGRAM



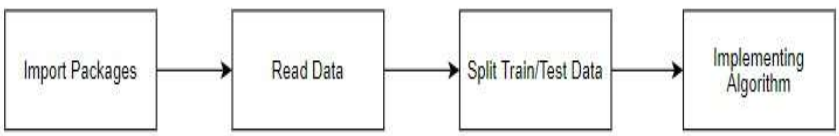
Voting Classifier:

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output. It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class. Voting Classifier supports two types of votings.

1. **Hard Voting:** In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the output class(A, A, B), so here the majority predicted A as output. Hence A will be the final prediction.
2. **Soft Voting**: In soft voting, the output class is the prediction based on the average of probability given to that class. Suppose given some input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B = (0.20, 0.32, 0.40). So the average for class A is 0.4333 and B is 0.3067, the winner is clearly class A because it had the highest probability averaged by each classifier.

```
vot_clf.fit(X_train, y_train)
pred_vtng = vot_clf.predict(X_test)
```

MODULE DIAGRAM



6.2 DEPLOYMENT:

In this model, we have used Random Forest algorithm for implementation. Above all the algorithms that have been implemented Random Forest algorithm has given the best accuracy rate.

Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees,* resulting in a *forest of trees,* hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.
The following are the basic steps involved in performing the random forest algorithm:

- Pick N random records from the dataset.
- Build a decision tree based on these N records.
- Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

Getting Accuracy

```
accuracy = accuracy_score(y_test,predicted_rf)
print('Accuracy of Random Forest Classifier is: ',accuracy*100)
```

```
Accuracy of Random Forest Classifier is:  96.1038961038961
```

CHAPTER 7
PERFORMANCE EVALUATION

7.1 Results & Discussion

In our system, we used a supervised machine learning algorithm having subcategories as classification and regression. The classification algorithm will be most suitable for this system. Finally we used the random forest algorithm and our model has showed the accuracy rate of 95.246%.

The output of the project is based on five types of CO2 emission ratings:

1. Very low rate
2. Moderate
3. Stage 1 High rate
4. Stage 2 High rate
5. Stage 3 High rate

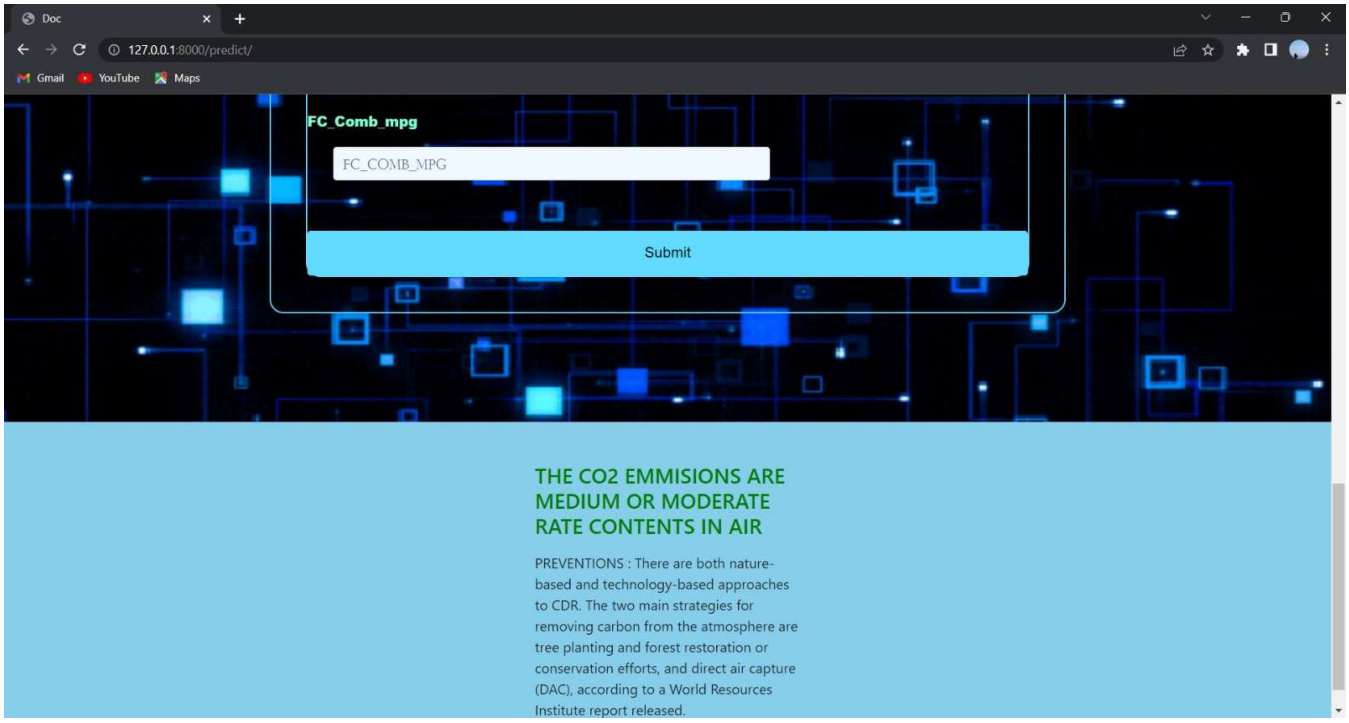


Fig 7.1: Output prdicting screen

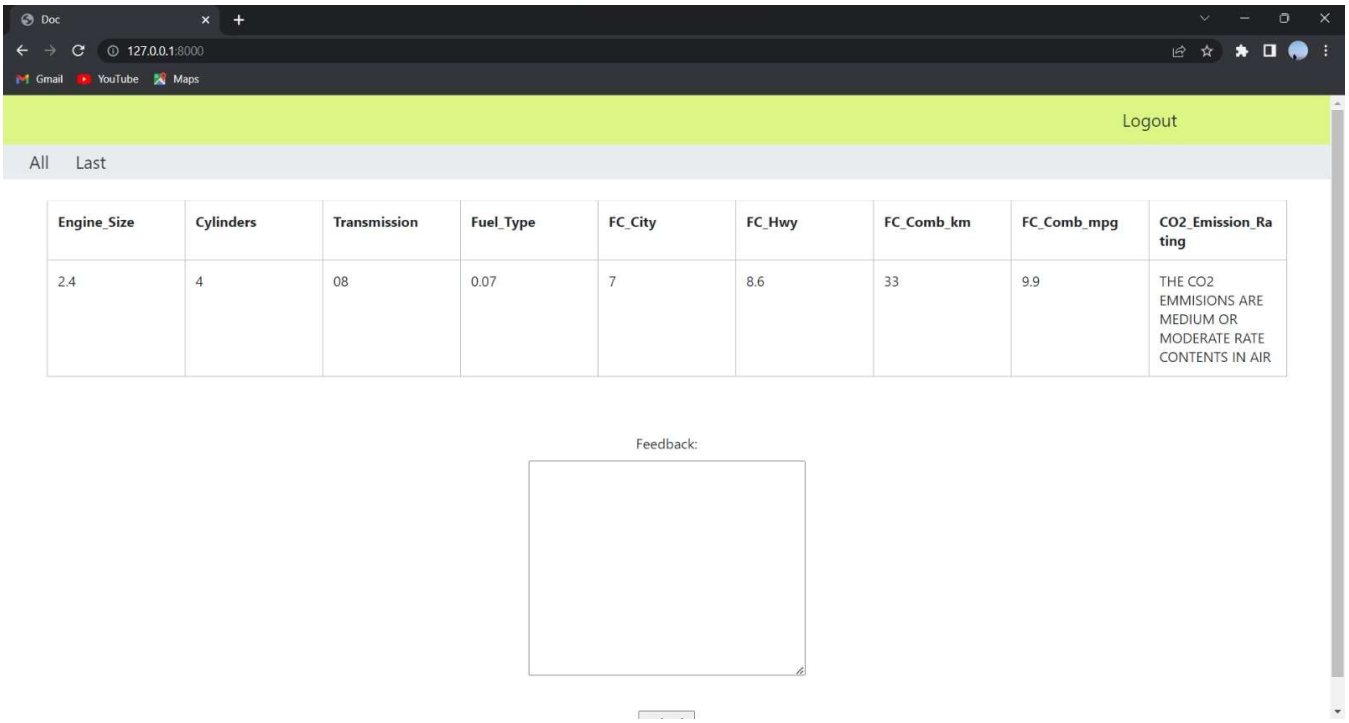


Fig 7.2: CO2 control inspector page

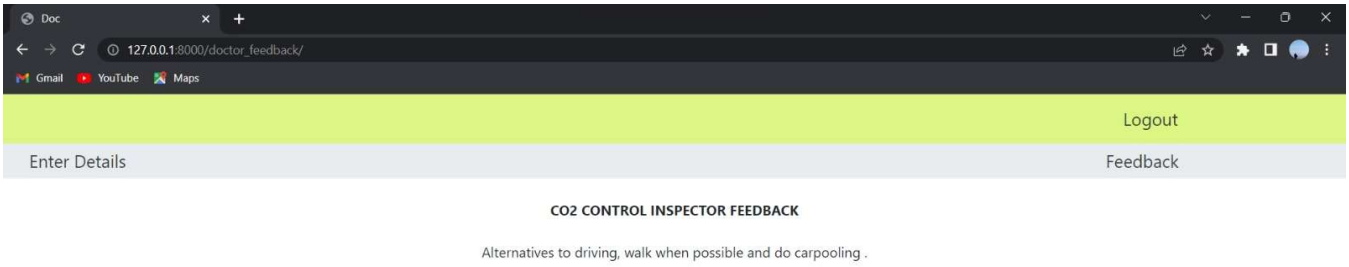


Fig 7.3: CO2 control inspector feeedback page

7.2 CLASSIFICATION REPORT:

```
cr = classification_report(y_test,predicted_ab)
print('Classification report\n---------------------\n',cr)
```

```
Classification report
---------------------
              precision    recall  f1-score   support

           0       0.87      0.67      0.76       153
           1       0.57      0.69      0.63       185
           2       0.74      0.73      0.73       230

    accuracy                           0.70       568
   macro avg       0.73      0.70      0.71       568
weighted avg       0.72      0.70      0.70       568
```

Fig 7.4: Classification Report of Adaboost algorithm.

```
cr = classification_report(y_test,predicted_xgb)
print('Classification report\n---------------------\n',cr)
```

```
Classification report
---------------------
              precision    recall  f1-score   support

           0       0.94      0.94      0.94        34
           1       0.97      0.95      0.96       119
           2       0.95      0.95      0.95       185
           3       0.94      0.95      0.94       113
           4       0.96      0.97      0.96       117

    accuracy                           0.95       568
   macro avg       0.95      0.95      0.95       568
weighted avg       0.95      0.95      0.95       568
```

Fig 7.5: Classification Report of XGboost algorithm

```
cr = classification_report(y_test,predicted_rf)
print('Classification report\n---------------------\n',cr)
```

```
Classification report
---------------------
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       185
           1       0.96      0.96      0.96       185
           2       0.93      0.91      0.92       184
           3       0.98      0.97      0.98       185
           4       0.95      0.97      0.96       185

    accuracy                           0.96       924
   macro avg       0.96      0.96      0.96       924
weighted avg       0.96      0.96      0.96       924
```

Fig 7.6: Classification report of Random Forest algorithm

```
cr = classification_report(y_test,pred_vtng)
print('Classification report\n---------------------\n',cr)
```

```
Classification report
---------------------
              precision    recall  f1-score   support

           0       0.99      0.92      0.95       153
           1       0.92      0.95      0.93       185
           2       0.94      0.96      0.95       230

    accuracy                           0.95       568
   macro avg       0.95      0.94      0.95       568
weighted avg       0.95      0.95      0.95       568
```

Fig 7.7: Classification report of Voting Classifier

7.3 COMPARISON OF ALGORITHMS:

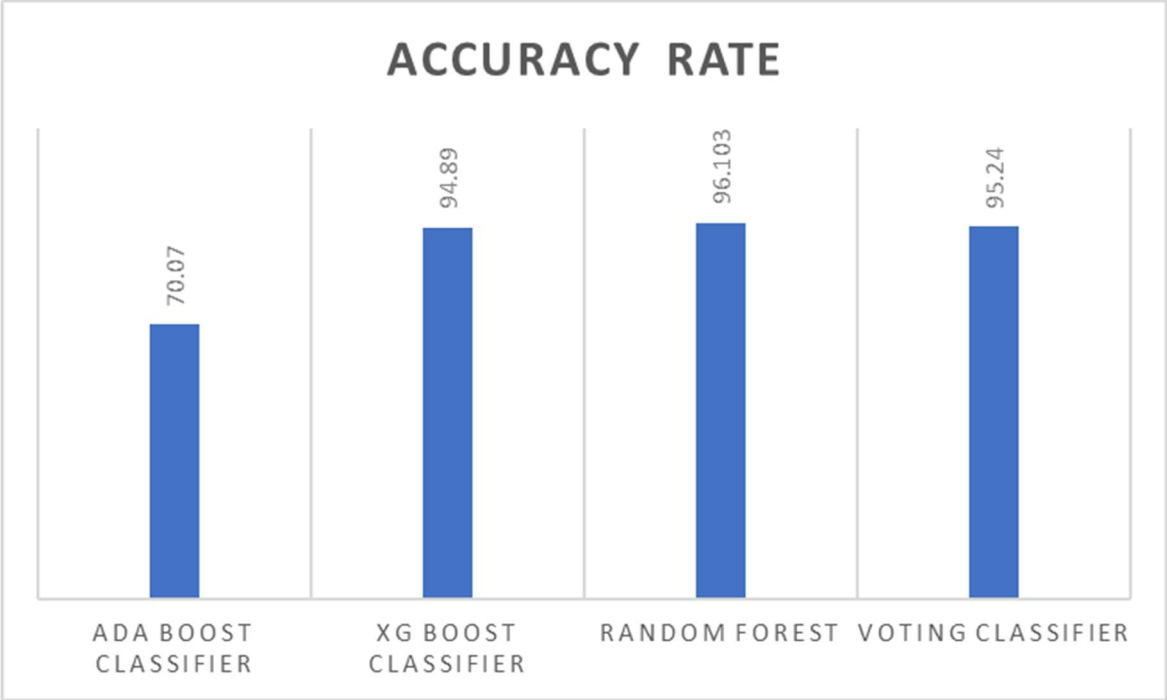


Fig 7.8: COMPARISON OF ALGORITHMS

The above graph shows the comparison between four algorithms of proposed work. From the above, it can be said that “Random Forest Algorithm” yields highest accuracy compared to other three.

CHAPTER 8
CONCLUSION

8.1 Conclusion and Future Enhancements:

Conclusion:

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set of higher accuracy score algorithm will be find out. The founded one is used in the application which can help to find the co2 emission of the vehicle. And it does suggest the way to reduce the CO2 emission rate.

Future Work:

- ❖ Deploying the project in the cloud.
- ❖ To optimize the work to implement in the IOT system.

APPENDICES

A. SAMPLE DATASET

Sl.No	Model Year	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fuel Consumption	Fuel Consum	Fuel Consum	Fuel Consump	CO2 Emission Rating
0	2022	Acura	ILX	Compact	2.4	4	AM8	Z	9.9	7	8.6	33	3
1	2022	Acura	MDX SH-AWD	SUV: Small	3.5	6	AS10	Z	12.6	9.4	11.2	25	5
2	2022	Acura	RDX SH-AWD	SUV: Small	2	4	AS10	Z	11	8.6	9.9	29	6
3	2022	Acura	RDX SH-AWD A-SPEC	SUV: Small	2	4	AS10	Z	11.3	9.1	10.3	27	6
4	2022	Acura	TLX SH-AWD	Compact	2	4	AS10	Z	11.2	8	9.8	29	7
5	2022	Acura	TLX SH-AWD A-SPEC	Compact	2	4	AS10	Z	11.3	8.1	9.8	29	7
6	2022	Acura	TLX Type S	Compact	3	6	AS10	Z	12.3	9.4	11	26	5
7	2022	Acura	TLX Type S (Performanc	Compact	3	6	AS10	Z	12.3	9.8	11.2	25	5
8	2022	Alfa Romeo	Giulia	Mid-size	2	4	A8	Z	10	7.2	8.7	32	3
9	2022	Alfa Romeo	Giulia AWD	Mid-size	2	4	A8	Z	10.5	7.7	9.2	31	3
10	2022	Alfa Romeo	Giulia Quadrifoglio	Mid-size	2.9	6	A8	Z	13.5	9.3	11.6	24	3
11	2022	Alfa Romeo	Stelvio	SUV: Small	2	4	A8	Z	10.3	8.1	9.3	30	3
12	2022	Alfa Romeo	Stelvio AWD	SUV: Small	2	4	A8	Z	10.8	8.3	9.6	29	3
13	2022	Alfa Romeo	Stelvio AWD Quadrifogl	SUV: Small	2.9	6	A8	Z	13.9	10.3	12.3	23	3
14	2022	Aston Martin	DB11 V8	Minicompact	4	8	A8	Z	13	9.8	11.5	25	5
15	2022	Aston Martin	DB11 V12	Minicompact	5.2	12	A8	Z	16.4	10.7	13.8	20	3
16	2022	Aston Martin	DBS V12	Minicompact	5.2	12	A8	Z	16.4	10.7	13.8	20	3
17	2022	Aston Martin	DBX V8	SUV: Standard	4	8	A9	Z	16.8	11.9	14.6	19	5
18	2022	Aston Martin	Vantage V8	Two-seater	4	8	A8	Z	13.1	9.6	11.5	25	5
19	2022	Audi	A3 Sedan 40 TFSI quattr	Subcompact	2	4	AM7	X	8.5	6.6	7.6	37	7
20	2022	Audi	A4 Sedan 40 TFSI quattr	Compact	2	4	AM7	Z	9.1	7	8.2	34	5
21	2022	Audi	A4 Sedan 45 TFSI quattr	Compact	2	4	AM7	Z	9.8	7.6	8.8	32	5
22	2022	Audi	A4 allroad 45 TFSI quatt	Station wagon:	2	4	AM7	Z	9.8	7.9	8.9	32	5
23	2022	Audi	A5 Cabriolet 45 TFSI qu	Subcompact	2	4	AM7	Z	10.4	7.5	9.1	31	5
24	2022	Audi	A5 Coupe 45 TFSI quatt	Subcompact	2	4	AM7	Z	9.8	7.6	8.8	32	5
25	2022	Audi	A5 Sportback 45 TFSI qu	Mid-size	2	4	AM7	Z	9.8	7.6	8.8	32	5
26	2022	Audi	A6 Sedan 45 TFSI quattr	Mid-size	2	4	AM7	Z	10.2	7.4	8.9	32	5
27	2022	Audi	A6 Sedan 55 TFSI quattr	Mid-size	3	6	AM7	Z	11.1	7.8	9.6	29	5
28	2022	Audi	A6 allroad 55 TFSI quatt	Station wagon:	3	6	AM7	Z	11.5	8.3	10	28	5
29	2022	Audi	A7 Sportback 55 TFSI qu	Mid-size	3	6	AM7	Z	11.1	7.8	9.6	29	5
30	2022	Audi	A8 L Sedan 55 TFSI quat	Full-size	3	6	AS8	Z	12.6	8.3	10.6	27	5
31	2022	Audi	Q3 40 TFSI quattro	SUV: Small	2	4	AS8	X	10.4	7.7	9.2	31	7
32	2022	Audi	Q3 45 TFSI quattro	SUV: Small	2	4	AS8	X	11.4	8.3	10	28	7
33	2022	Audi	Q5 40 TFSI quattro	SUV: Small	2	4	AM7	Z	10.3	8.1	9.3	30	5
34	2022	Audi	Q5 45 TFSI quattro	SUV: Small	2	4	AM7	Z	10.3	8.4	9.4	30	5
35	2022	Audi	Q5 Sportback 45 TFSI qu	SUV: Small	2	4	AM7	Z	10.3	8.4	9.4	30	5
36	2022	Audi	Q7 45 TFSI quattro	SUV: Standard	2	4	AS8	Z	12	9.4	10.8	26	3
37	2022	Audi	Q7 55 TFSI quattro	SUV: Standard	3	6	AS8	Z	12.8	10.5	11.7	24	5
38	2022	Audi	Q8 55 TFSI quattro	SUV: Standard	3	6	AS8	Z	12.8	10.5	11.7	24	5
39	2022	Audi	R8 Coupe Performance	Two-seater	5.2	10	AM7	Z	16.7	10.3	13.8	20	1
40	2022	Audi	R8 Coupe Performance	Two-seater	5.2	10	AM7	Z	17.9	12.1	15.3	18	1
41	2022	Audi	R8 Spyder Performance	Two-seater	5.2	10	AM7	Z	16.7	10.3	13.8	20	1
42	2022	Audi	R8 Spyder Performance	Two-seater	5.2	10	AM7	Z	17.9	12.1	15.3	18	1
43	2022	Audi	RS 5 Coupe quattro	Subcompact	2.9	6	AS8	Z	13	9.4	11.4	25	5
44	2022	Audi	RS 5 Sportback quattro	Mid-size	2.9	6	AS8	Z	13.1	9.4	11.4	25	5
45	2022	Audi	RS 6 Avant quattro	Station wagon:	4	8	AS8	Z	16.1	10.7	13.7	21	3
46	2022	Audi	RS 7 Sportback quattro	Mid-size	4	8	AS8	Z	16	10.5	13.5	21	3
47	2022	Audi	RS Q8 quattro	SUV: Standard	4	8	AS8	Z	18	12.3	15.4	18	3
48	2022	Audi	S3 Sedan quattro	Subcompact	2	4	AM7	Z	10	7.2	8.8	32	5
49	2022	Audi	S4 Sedan quattro	Compact	3	6	AS8	Z	11.1	8	9.7	29	5
50	2022	Audi	S5 Cabriolet quattro	Subcompact	3	6	AS8	Z	11.3	8.4	10	28	5

Fig A.1: Sample Dataset

The above sample screenshot is the dataset that has been used in the model in order to implement the model.

B. SAMPLE CODING

MODULE 1

DATA PRE-PROCESSING:

```
import pandas as p
import numpy as n
import warnings
warnings.filterwarnings('ignore')
data=p.read_csv('CO2.csv')
data.head()
data.shape
data.isnull()
data.columns
data['Make'].unique()
data['Make'].nunique()
data['Vehicle Class'].unique()
data['Fuel Type'].unique()
data['CO2 Emission Rating'].unique()
data.tail(10)
data.columns
df = data.dropna()
df.shape
df.isnull().sum()
df.describe()
df.corr()
df.info()
p.crosstab(df["Fuel Type"], df["CO2 Emission Rating"])
df["Vehicle Class"].value_counts()
df["Make"].value_counts()
p.Categorical(df["Cylinders"]).describe()
p.Categorical(df["Fuel Type"]).describe()
df.duplicated()
sum(df.duplicated())
df
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
var = ['Model Year', 'Make', 'Model', 'Vehicle Class', 'Transmission', 'Fuel Type']

for i in var:
    df[i] = le.fit_transform(df[i]).astype(int)
df
```

MODULE 2

DATA VISUALIZATION:

```
import pandas as p
import numpy as n
import seaborn as s
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')

data=p.read_csv('CO2.csv')

df=data.dropna()

df.columns

df = df.rename({'Model Year': 'Model_Year', 'Vehicle Class': 'Vehicle_Class', 'Engine Size(L)': 'Engine_Size',
                'Fuel Type': 'Fuel_Type', 'Fuel Consumption (City (L/100 km)': 'FC_City',
                'Fuel Consumption(Hwy (L/100 km))': 'FC_Hwy', 'Fuel Consumption(Comb (L/100 km))':
'FC_Comb_km',
                'Fuel Consumption(Comb (mpg))': 'FC_Comb_mpg', 'CO2 Emission Rating': 'CO2_Emission_Rating'},
axis=1)

df.columns

plt.figure(figsize=(16,4))
plt.subplot(1,2,1)
plt.hist(df['FC_City'])
plt.title("Mileage at City")
plt.xlabel("Mileage")
plt.ylabel("Number of Vehicles")
plt.subplot(1,2,2)
plt.hist(df['FC_Hwy'])
plt.xlabel('Mileage')
plt.title('Mileage at Highway')
plt.ylabel("Number of Vehicles")
plt.show()

plt.figure(figsize=(8,4))
plt.scatter(df["Fuel_Type"],df["CO2_Emission_Rating"],color="red")
plt.xlabel('Fuel Types')
plt.ylabel('CO2 Rating')

plt.figure(figsize= (10,4))
plt.hist(df['Engine_Size'],color="m")
plt.xlabel("Engine Size")
```

```
plt.ylabel("Number of Vehicles")
plt.show()

import seaborn as s
s.boxplot(df['FC_City'], color='y')

import seaborn as s
s.boxplot(df['FC_Comb_mpg'], color='g')

#Density Plot
plt.figure(figsize= (6,4))
df["CO2_Emission_Rating"].plot(kind='density')
plt.show()

fig, ax = plt.subplots(figsize=(15,7))
s.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'coolwarm',ax=ax)


def Contract(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(7,7), autopct='%1.2f%%', fontsize = 10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return n.round(dataframe_pie/df.shape[0]*100,2)
Contract(df, 'CO2_Emission_Rating')
```

Module – 3

IMPLEMENTING ADABOOST ALGORITHM

```
#import library packages
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

# Load given dataset
data = pd.read_csv("CO2.csv")

df = data.dropna()

df = df.rename({'Model Year': 'Model_Year', 'Vehicle Class': 'Vehicle_Class', 'Engine Size(L)': 'Engine_Size',
                'Fuel Type': 'Fuel_Type', 'Fuel Consumption (City (L/100 km)': 'FC_City',
                'Fuel Consumption(Hwy (L/100 km))': 'FC_Hwy', 'Fuel Consumption(Comb (L/100 km))': 'FC_Comb_km',
                'Fuel Consumption(Comb (mpg))': 'FC_Comb_mpg', 'CO2 Emission Rating': 'CO2_Emission_Rating'},
axis=1)

df['CO2_Emission_Rating'].value_counts()

df['CO2_Emission_Rating'] = df.CO2_Emission_Rating.map({1:0,3:0,5:1,6:2,7:2})

df['CO2_Emission_Rating'].value_counts()

df.columns

del df['Sl.No']
del df['Model_Year']
del df['Make']
del df['Model']
del df['Vehicle_Class']
```

```
df.info()
```

```
from sklearn.preprocessing import LabelEncoder
col = ['Transmission', 'Fuel_Type']
label = LabelEncoder()
for i in col:
    df[i] = label.fit_transform(df[i]).astype(int)
```

```
df.head()
```

```
inputs = df.drop(labels='CO2_Emission_Rating', axis=1)
output = df.loc[:,'CO2_Emission_Rating']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, output, test_size=0.3, random_state=1, stratify=output)
print("Number of Training Dataset: ", len(X_train))
print("Number of Testing Dataset: ", len(X_test))
print("Total Number of Dataset: ", len(X_train)+len(X_test))
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, plot_confusion_matrix
```

Training Process

```
ab = AdaBoostClassifier()
ab.fit(X_train,y_train)
predicted_ab = ab.predict(X_test)
```

Getting Accuracy

```
accuracy = accuracy_score(y_test,predicted_ab)
print('Accuracy of AdaBoost Classifier is: ',accuracy*100)
```

Finding Clasiification Report

```
cr = classification_report(y_test,predicted_ab)
print('Classification report\n----------------------\n',cr)
```

Finding Confusion Matrix

```
cm = confusion_matrix(y_test,predicted_ab)
print('Confusion matrix\n----------------\n',cm)
```

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(7,7))
plot_confusion_matrix(ab, X_test, y_test, ax=ax)
plt.title('Confusion Matrix of AdaBoost Classifier')
plt.show()
```

```
DF = pd.DataFrame()
DF["y_test"] = y_test
DF["predicted"] = predicted_ab
DF.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(DF["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(DF["y_test"][:100],  marker='o', linestyle='dashed', color='green')
plt.show()
```

Module – 4

IMPLEMENTING XGBOOST ALGORITHM

```
#import library packages
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

# Load given dataset
data = pd.read_csv("CO2.csv")

df = data.dropna()

df = df.rename({'Model Year': 'Model_Year', 'Vehicle Class': 'Vehicle_Class', 'Engine Size(L)': 'Engine_Size',
                'Fuel Type': 'Fuel_Type', 'Fuel Consumption (City (L/100 km)': 'FC_City',
                'Fuel Consumption(Hwy (L/100 km))': 'FC_Hwy', 'Fuel Consumption(Comb (L/100 km))': 'FC_Comb_km',
                'Fuel Consumption(Comb (mpg))': 'FC_Comb_mpg', 'CO2 Emission Rating': 'CO2_Emission_Rating'},
axis=1)

df['CO2_Emission_Rating'].value_counts()

df['CO2_Emission_Rating'] = df.CO2_Emission_Rating.map({1:0,3:0,5:1,6:2,7:2})

df['CO2_Emission_Rating'].value_counts()

df.columns

del df['Sl.No']
del df['Model_Year']
del df['Make']
del df['Model']
del df['Vehicle_Class']

df.info()

from sklearn.preprocessing import LabelEncoder
col = ['Transmission', 'Fuel_Type']
label = LabelEncoder()
for i in col:
    df[i] = label.fit_transform(df[i]).astype(int)

df.head()

inputs = df.drop(labels='CO2_Emission_Rating', axis=1)
output = df.loc[:,'CO2_Emission_Rating']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, output, test_size=0.3, random_state=1, stratify=output)
print("Number of Training Dataset: ", len(X_train))
print("Number of Testing Dataset: ", len(X_test))
print("Total Number of Dataset: ", len(X_train)+len(X_test))

XGBoost Algorithm

import xgboost as xg
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, plot_confusion_matrix

Training Process

xgb = xg.XGBClassifier()
xgb.fit(X_train,y_train)
predicted_xgb = xgb.predict(X_test)
```

Getting Accuracy

```
accuracy = accuracy_score(y_test,predicted_xgb)
print('Accuracy of XGBoost Classifier is: ',accuracy*100)
```

Finding Clasiification Report

```
cr = classification_report(y_test,predicted_xgb)
print('Classification report\n----------------------\n',cr)
```

Finding Confusion Matrix

```
cm = confusion_matrix(y_test,predicted_xgb)
print('Confusion matrix\n----------------\n',cm)
```

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(7,7))
plot_confusion_matrix(xgb, X_test, y_test, ax=ax)
plt.title('Confusion Matrix of XGBBoost Classifier')
plt.show()
```

```
DF = pd.DataFrame()
DF["y_test"] = y_test
DF["predicted"] = predicted_xgb
DF.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(DF["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(DF["y_test"][:100],  marker='o', linestyle='dashed', color='green')
plt.show()
```

Module – 5

IMPLEMENTING RANDOM FOREST CLASSIFIER

```
#import library packages
import pandas as pd
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv("CO2.csv")
```

```
df = data.dropna()
```

```
df = df.rename({'Model Year': 'Model_Year', 'Vehicle Class': 'Vehicle_Class', 'Engine Size(L)': 'Engine_Size',
                'Fuel Type': 'Fuel_Type', 'Fuel Consumption (City (L/100 km)': 'FC_City',
                'Fuel Consumption(Hwy (L/100 km))': 'FC_Hwy', 'Fuel Consumption(Comb (L/100 km))': 'FC_Comb_km',
                'Fuel Consumption(Comb (mpg))': 'FC_Comb_mpg', 'CO2 Emission Rating': 'CO2_Emission_Rating'},
axis=1)
```

```
df['CO2_Emission_Rating'].value_counts()
```

```
df['CO2_Emission_Rating'] = df.CO2_Emission_Rating.map({1:0,3:0,5:1,6:2,7:2})
```

```
df['CO2_Emission_Rating'].value_counts()
```

```
df.columns
```

```
del df['Sl.No']
del df['Model_Year']
del df['Make']
```

```
del df['Model']
del df['Vehicle_Class']
```

```
df.info()
```

```
from sklearn.preprocessing import LabelEncoder
col = ['Transmission', 'Fuel_Type']
label = LabelEncoder()
for i in col:
    df[i] = label.fit_transform(df[i]).astype(int)
```

```
df.head()
```

```
inputs = df.drop(labels='CO2_Emission_Rating', axis=1)
output = df.loc[:,'CO2_Emission_Rating']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, output, test_size=0.3, random_state=1, stratify=output)
print("Number of Training Dataset: ", len(X_train))
print("Number of Testing Dataset: ", len(X_test))
print("Total Number of Dataset: ", len(X_train)+len(X_test))
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, plot_confusion_matrix
```

Training Process

```
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
predicted_rf = rf.predict(X_test)
```

Getting Accuracy

```
accuracy = accuracy_score(y_test,predicted_rf)
print('Accuracy of Random Forest Classifier is: ',accuracy*100)
```

Finding Clasiification Report

```
cr = classification_report(y_test,predicted_rf)
print('Classification report\n----------------------\n',cr)
```

Finding Confusion Matrix

```
cm = confusion_matrix(y_test,predicted_rf)
print('Confusion matrix\n----------------\n',cm)
```

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(7,7))
plot_confusion_matrix(rf, X_test, y_test, ax=ax)
plt.title('Confusion Matrix of Random Forest Classifier')
plt.show()
```

```
DF = pd.DataFrame()
DF["y_test"] = y_test
DF["predicted"] = predicted_rf
DF.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(DF["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(DF["y_test"][:100],  marker='o', linestyle='dashed', color='green')
```

```
plt.show()
```

Saving Model

```
import joblib
joblib.dump(rf,'RFC.pkl')
```

Module – 6

IMPLEMENTING VOTING CLASSIFIER ALGORITHM

```
#import library packages
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
# Load given dataset
data = pd.read_csv("CO2.csv")
df = data.dropna()
df = df.rename({'Model Year': 'Model_Year', 'Vehicle Class': 'Vehicle_Class', 'Engine Size(L)': 'Engine_Size',
            'Fuel Type': 'Fuel_Type', 'Fuel Consumption (City (L/100 km)': 'FC_City',
            'Fuel Consumption(Hwy (L/100 km))': 'FC_Hwy', 'Fuel Consumption(Comb (L/100 km))': 'FC_Comb_km',
            'Fuel Consumption(Comb (mpg))': 'FC_Comb_mpg', 'CO2 Emission Rating': 'CO2_Emission_Rating'},
axis=1)
df['CO2_Emission_Rating'].value_counts()
df['CO2_Emission_Rating'] = df.CO2_Emission_Rating.map({1:0,3:0,5:1,6:2,7:2})
df['CO2_Emission_Rating'].value_counts()
df.columns
del df['Sl.No']
del df['Model_Year']
del df['Make']
del df['Model']
del df['Vehicle_Class']
df.info()
from sklearn.preprocessing import LabelEncoder
col = ['Transmission', 'Fuel_Type']
label = LabelEncoder()
for i in col:
    df[i] = label.fit_transform(df[i]).astype(int)
df.head()
inputs = df.drop(labels='CO2_Emission_Rating', axis=1)
output = df.loc[:,'CO2_Emission_Rating']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, output, test_size=0.3, random_state=1, stratify=output)
print("Number of Training Dataset: ", len(X_train))
print("Number of Testing Dataset: ", len(X_test))
print("Total Number of Dataset: ", len(X_train)+len(X_test))
```

Voting Classifier

```
from sklearn.ensemble import AdaBoostClassifier
import xgboost as xgb
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, plot_confusion_matrix

ab = AdaBoostClassifier()
xg = xgb.XGBClassifier()
rf = RandomForestClassifier()

vot_clf = VotingClassifier(estimators=[('AdaBoost', ab), ('XGBoost', xg), ('RandomForest', rf)], voting='hard')

vot_clf.fit(X_train, y_train)
pred_vtng = vot_clf.predict(X_test)

Finding Accuracy

accuracy = accuracy_score(y_test,pred_vtng)
print('Accuracy of Voting Classifier is: ',accuracy*100)

Finding Classification Report

cr = classification_report(y_test,pred_vtng)
print('Classification report\n----------------------\n',cr)

Finding Confusion Matrix

cm = confusion_matrix(y_test,pred_vtng)
print('Confusion matrix\n----------------\n',cm)

import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(7,7))
plot_confusion_matrix(vot_clf, X_test, y_test, ax=ax)
plt.title('Confusion Matrix of Voting Classifier')
plt.show()

DF = pd.DataFrame()
DF["y_test"] = y_test
DF["predicted"] = pred_vtng
DF.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(DF["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(DF["y_test"][:100],  marker='o', linestyle='dashed', color='green')
plt.show()
```

B.1 SAMPLE SCREENS

```
data.columns
```

```
Index(['Sl.No', 'Model Year', 'Make', 'Model', 'Vehicle Class',
       '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))',
       'CO2 Emission Rating'],
      dtype='object')
```

```
df = data.dropna()
```

```
df.shape
```

```
(1892, 14)
```

```
df.isnull().sum()
```

```
Sl.No                              0
Model Year                         0
Make                               0
Model                              0
Vehicle Class                      0
Engine Size(L)                     0
Cylinders                          0
Transmission                       0
Fuel Type                          0
```

Fig B.1.1: Data pre-processing coding.

```
def Contract(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(7,7), autopct='%1.2f%%', fontsize = 10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return n.round(dataframe_pie/df.shape[0]*100,2)
Contract(df, 'CO2_Emission_Rating')
```

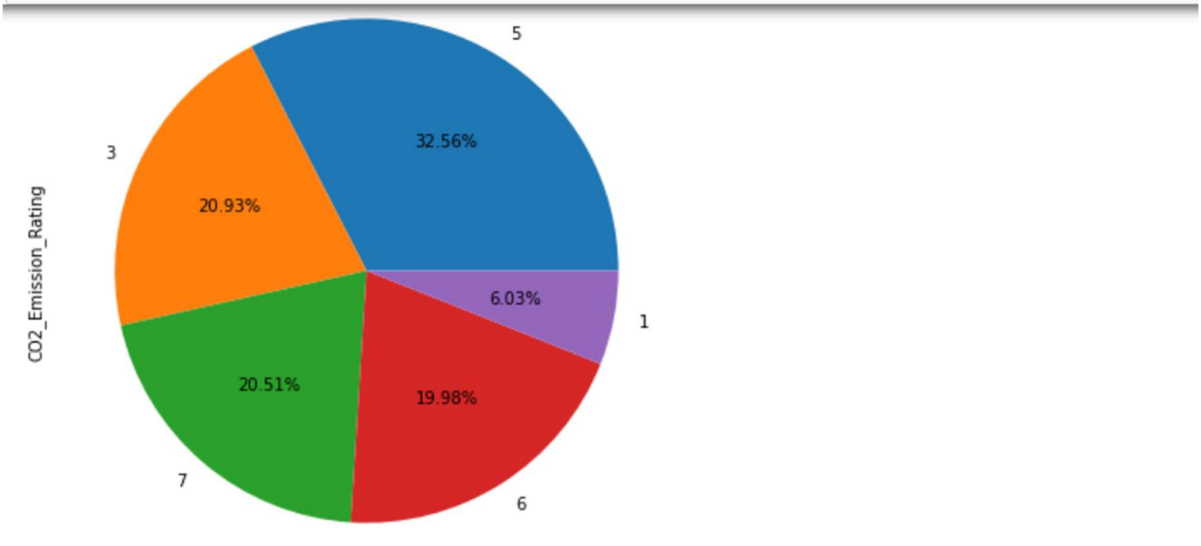


Fig B.1.2: Data visualization coding

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, output, test_size=0.3, random_state=1, stratify=output)
print("Number of Training Dataset: ", len(X_train))
print("Number of Testing Dataset: ", len(X_test))
print("Total Number of Dataset: ", len(X_train)+len(X_test))
```

```
Number of Training Dataset:  1324
Number of Testing Dataset:  568
Total Number of Dataset:  1892
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, plot_confusion_matrix
```

Training Process

```
ab = AdaBoostClassifier()
ab.fit(X_train,y_train)
predicted_ab = ab.predict(X_test)
```

Fig B.1.3: Implementing Adaboost Classifier

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, output, test_size=0.3, random_state=1, stratify=output)
print("Number of Training Dataset: ", len(X_train))
print("Number of Testing Dataset: ", len(X_test))
print("Total Number of Dataset: ", len(X_train)+len(X_test))
```

```
Number of Training Dataset:  1324
Number of Testing Dataset:  568
Total Number of Dataset:  1892
```

XGBoost Algorithm

```
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, plot_confusion_matrix
```

Training Process

```
xgb = XGBClassifier()
xgb.fit(X_train,y_train)
predicted_xgb = xgb.predict(X_test)
```

Fig B.1.4: Implementing XGboost algorithm

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1, stratify=y)
print("Number of Training Dataset: ", len(X_train))
print("Number of Testing Dataset: ", len(X_test))
print("Total Number of Dataset: ", len(X_train)+len(X_test))
```

```
Number of Training Dataset:  2156
Number of Testing Dataset:  924
Total Number of Dataset:  3080
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

Training Process

```
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
predicted_rf = rf.predict(X_test)
```

Fig B.1.5: Implementing Random Forest algorithm

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, output, test_size=0.3, random_state=1, stratify=output)
print("Number of Training Dataset: ", len(X_train))
print("Number of Testing Dataset: ", len(X_test))
print("Total Number of Dataset: ", len(X_train)+len(X_test))
```

```
Number of Training Dataset:  1324
Number of Testing Dataset:  568
Total Number of Dataset:  1892
```

Voting Classifier

```
from sklearn.ensemble import AdaBoostClassifier
import xgboost as xgb
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, plot_confusion_matrix
```

```
ab = AdaBoostClassifier()
xg = xgb.XGBClassifier()
rf = RandomForestClassifier()
```

```
vot_clf = VotingClassifier(estimators=[('AdaBoost', ab), ('XGBoost', xg), ('RandomForest', rf)], voting='hard')
```

Fig B.1.6: Implementing Voting Classifier

C. SCREENSHOTS

```
accuracy = accuracy_score(y_test,predicted_ab)
print('Accuracy of AdaBoost Classifier is: ',accuracy*100)
```

```
Accuracy of AdaBoost Classifier is:  70.07042253521126
```

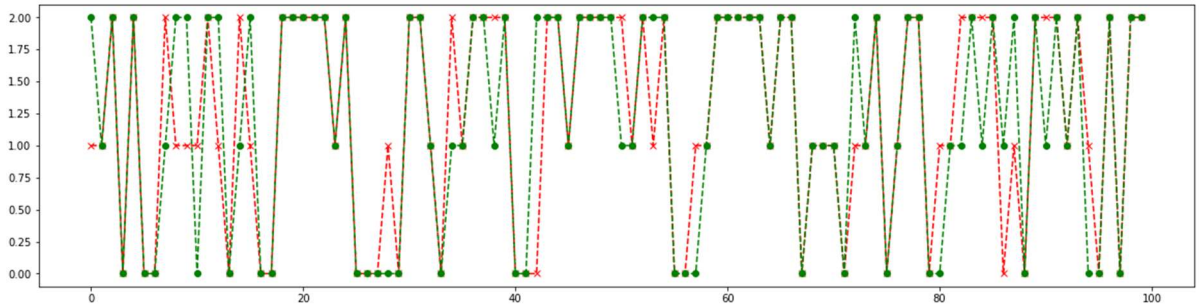


Fig C.1: Screenshot of Accuracy and Accuracy graph of AdaBoost algorithm

```
accuracy = accuracy_score(y_test,predicted_xgb)
print('Accuracy of XGBoost Classifier is: ',accuracy*100)
```

```
Accuracy of XGBoost Classifier is:  94.8943661971831
```

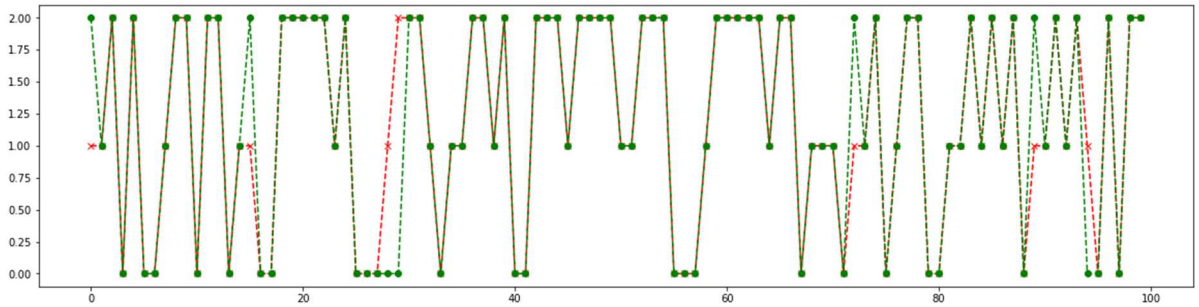


Fig C.2: Screenshot of Accuracy and Accuracy graph of XGBoost Classifier

```
accuracy = accuracy_score(y_test,predicted_rf)
print('Accuracy of Random Forest Classifier is: ',accuracy*100)
```

```
Accuracy of Random Forest Classifier is:  95.24647887323944
```

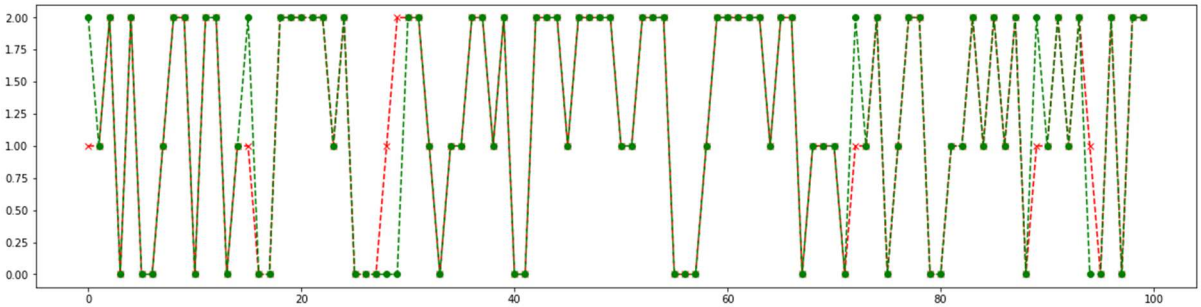


Fig C.3: Screenshot of Accuracy and Accuracy graph of Random Forest algorithm

```
accuracy = accuracy_score(y_test,pred_vtng)
print('Accuracy of Voting Classifier is: ',accuracy*100)
```

```
Accuracy of Voting Classifier is:  94.54225352112677
```

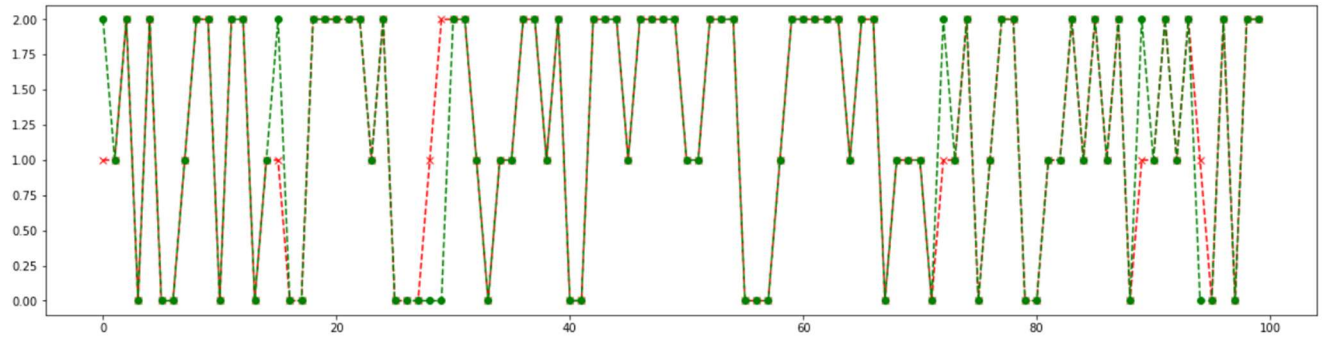


Fig C.4: Screenshot of Accuracy and Accuracy graph of Voting Classifier

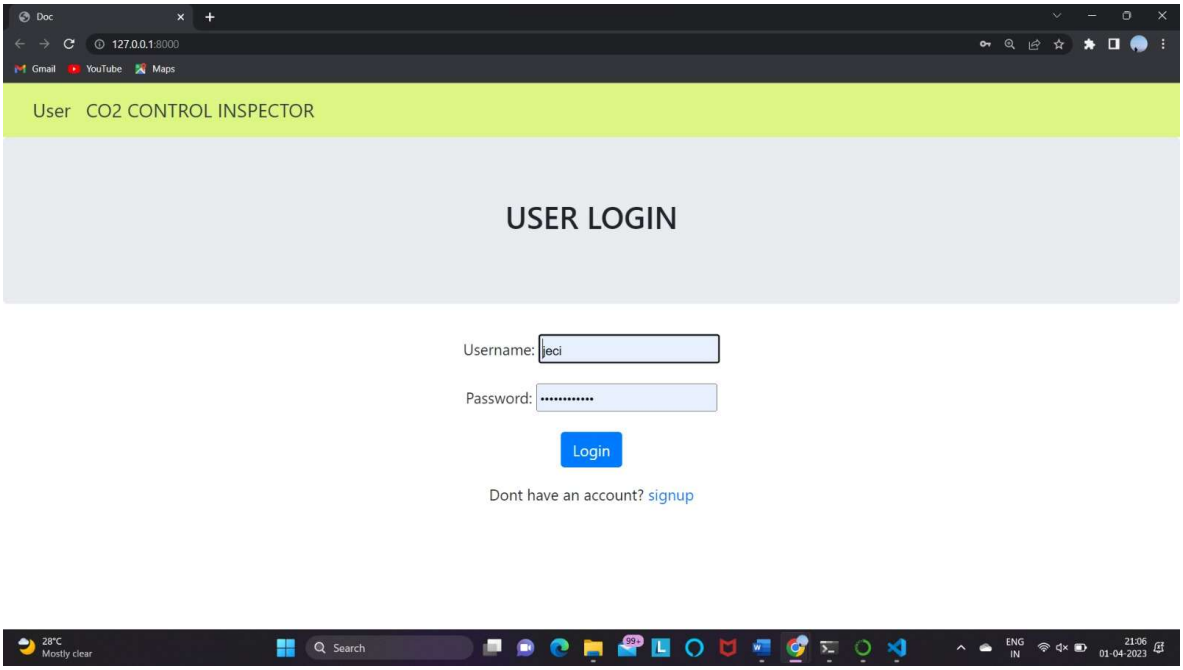


Fig C.5: User Login page

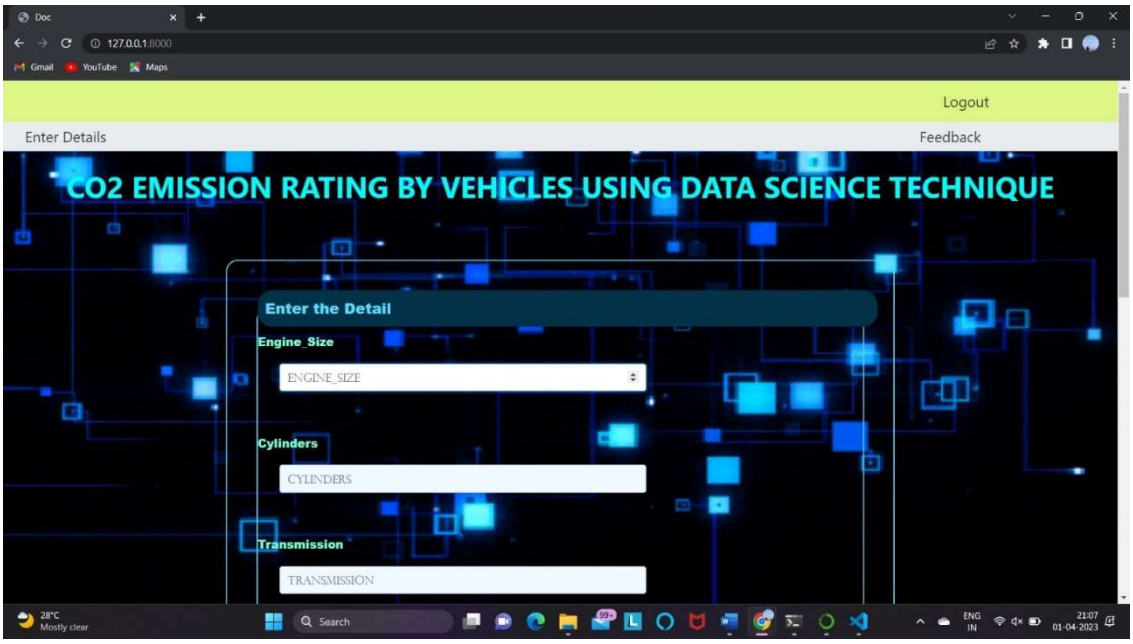


Fig C.6: Home page

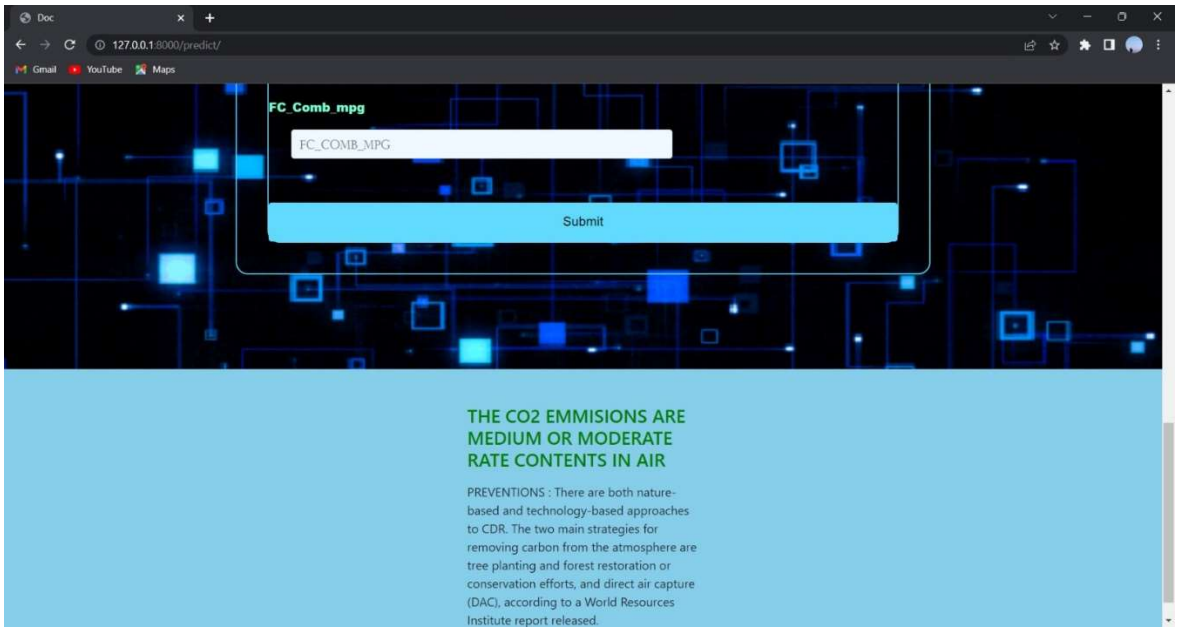


Fig C.7: Output Prediction page

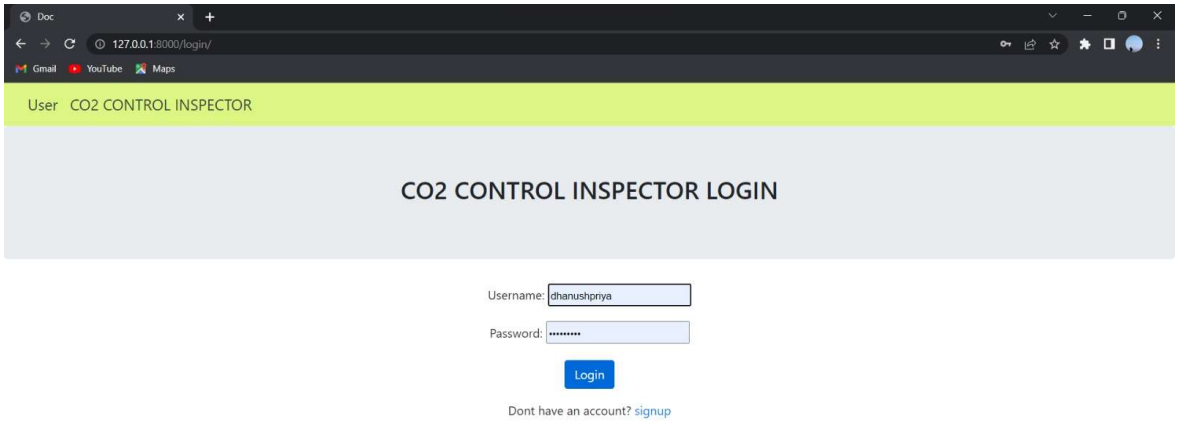


Fig C.8: CO2 Control Inspector Login page

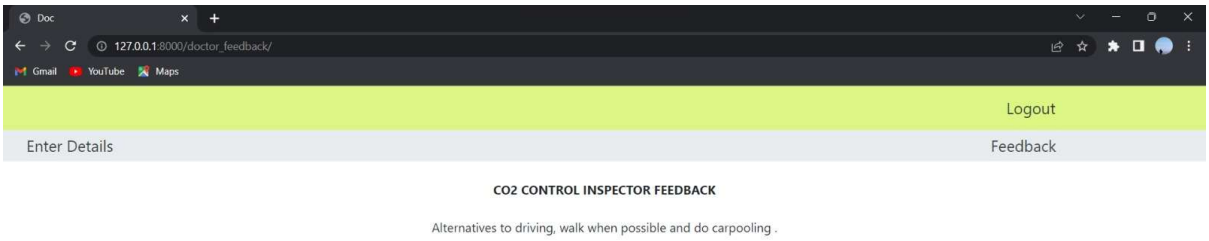


Fig C.9: CO2 Control Inspector Feedback page

9.References

[1] T. Jia, P. Zhang and B. Chen, "A Microscopic Model of Vehicle CO_2 Emissions Based on Deep Learning—A Spatiotemporal Analysis of Taxicabs in Wuhan, China," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 10, pp. 18446-18455, Oct.2022, doi: 10.1109/TITS.2022.3151655.

[2] Le Huy Hien, Ngo & Kor, Ah-Lian. (2022). Analysis and Prediction Model of Fuel Consumption and Carbon Dioxide Emissions of Light-Duty Vehicles. Applied Sciences. 12. 803.10.3390/app12020803.

[3] P. Kadam and S. Vijayumar, "Prediction Model: CO2 Emission Using Machine Learning," 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 2018, pp. 1-3, doi:10.1109/I2CT.2018.8529498.

[4] Huang, Huafang & Wu, Xiaomao & Cheng, Xianfu. (2021). The Prediction of Carbon Emission Information in Yangtze River Economic Zone by Deep Learning. Land. 10. 1380.10.3390/land10121380.

[5] Saleh, Chairul & Dzakiyullah, Nur & Nugroho, Jonathan. (2016). Carbon dioxide emission prediction using support vector machine. IOP Conference Series: Materials Science and Engineering. 114. 012148. 10.1088/1757-899X/114/1/012148.

[6] Kumar, Sandeep & Muhuri, Pranab. (2020). A Novel GDP Prediction Technique based on Transfer Learning using CO2 Emission Dataset.

[7] Jaworski, Artur & Mądziel, Maksymilian & Lejda, Kazimierz.(2019). Creating an emission model based on portable emission measurement system for the purpose of a roundabout. Environmental Science and Pollution Research. 26.10.1007/s11356-019-05264-1.

[8] Zhang, Jinlei & Chen, Feng & Wang, Zijia & Wang, Rui & Shi, Shunwei. (2018). Spatiotemporal Patterns of Carbon Emissions and Taxi Travel Using GPS Data in Beijing. Energies. 11. 500.10.3390/en11030500.

[9] L. Amarpuri, N. Yadav, G. Kumar and S. Agrawal, "Prediction of CO2 emissions using deep learning hybrid approach: A Case Study in Indian Context," 2019 Twelfth International Conference on Contemporary Computing (IC3), Noida, India, 2019, pp. 1-6, doi: 10.1109/IC3.2019.8844902.

[10] T. M. O. Santos, J. N. O. Júnior, M. Bessani and C. D. Maciel, "CO2 Emissions Forecasting in Multi-Source Power Generation Systems Using Dynamic Bayesian Network," 2021 IEEE International Systems Conference (SysCon), Vancouver, BC, Canada, 2021, pp. 1-8, doi: 10.1109/SysCon48628.2021.9447104.

[11] J. Ju and D. F. Kocaoglu, "Assessment on carbon capture technology: A literature review," Proceedings of PICMET '14 Conference: Portland International Center for Management of Engineering and Technology; Infrastructure and Service Integration, Kanazawa, Japan, 2014, pp. 484-490.

[12] Z. -z. Wang, L. -c. Fan and H. Mark, "Life-cycle assessment of CO2 emissions of buildings," 2011 International Conference on Remote Sensing, Environment and Transportation Engineering, Nanjing, China, 2011, pp. 438-441, doi: 10.1109/RSETE.2011.5964307.
[13] Hannah Ritchie, Max Roser and Pablo Rosado (2020) - "CO_2 and Greenhouse Gas Emissions". Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/co2-and-greenhouse-gas-emissions'[Online Resource]
[14] Szetela B, Majewska A, Jamroz P, Djalilov B and Salahodjaev R (2022) Renewable Energy and CO 2 Emissions in Top Natural Resource Rents Depending Countries: The Role of Governance. Front. Energy Res. 10:872941. doi:10.3389/fenrg.2022.872941.
[15] Ma N, Shum WY, Han T and Lai F (2021) Can Machine Learning be Applied to Carbon Emissions Analysis: An Application to the CO 2 Emissions Analysis Using Gaussian Process Regression. Front. Energy Res. 9:756311. doi: 10.3389/fenrg.2021.756311