A PROJECT REPORT ON

The Predicting Accident Risk Scores by Postal Codes through Machine Learning paradigms in societal analysis for Enhancing Road Safety

Major project submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY
(2020-2024)
BY

A. BINDU	**20241A1202**
D. SAI SHREEYA	**20241A1212**
K. TANMAI	**20241A1247**

Under the Esteemed guidance
of
V. Sailaja
Assistant Prof
Dept of IT.

DEPARTMENT OF INFORMATION TECHNOLOGY
GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS), HYDERABAD

CERTIFICATE

This is to certify that it is a bonafide record of Major Project work entitled **"**The Predicting Accident Risk Scores by Postal Codes through Machine Learning paradigms in societal analysis for Enhancing Road Safety**"** done by **A. BINDU(20241A1202), D. SAI SHREEYA(20241A1212), K. TANMAI(20241A1247),** of **B.Tech (IT)** in the Department of Information Technology, **Gokaraju Rangaraju Institute of Engineering and Technology** during the period 2020-2024 in the partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

V.Sailaja,
Assistant prof
(Internal project guide)

Dr Y J Nagendra Kumar,
Head of the Department,

(Project External)

ACKNOWLEDGEMENT

We take immense pleasure in expressing gratitude to our Internal guide**, V.Sailaja, Assistant Prof, Dept. of IT**, GRIET. We express our sincere thanks for her encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. Y J Nagendra Kumar,** HOD IT **,** our Project Co-ordinators**, Mr. G Vijendar Reddy , Mrs. L Sukanya** and **Mr. Y Subbarayudu** for their constant support during the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET**,** and **Dr. J. Praveen,** Principal, GRIET**,** for providing us the conductive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.

Name: Amaravadi Bindu
Email: amaravadibindu@gmail.com
Address: Madhira, Khammam
Contact No: 8790224972

Name: Digumarti Sai Shreeya
Email: digumartishreeya@gmail.com
Address: Nizampet, Hyderabad
Contact No: 8500969037

Name: Tanmai Kadaru
Email: tanmaikadaru@gmail.com
Address: Kompally, Hyderabad
Contact No: 7207879043

DECLARATION

This is to certify that the project entitled "**THE PREDICTING ACCIDENT RISK SCORES BY POSTAL CODES THROUGH MACHINE LEARNING PARADIGMS IN SOCIETAL ANALYSIS FOR ENHANCING ROAD SAFETY"** is a bonafide work done by us in partial fulfillment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in the Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

A. BINDU **20241A1202**

D. SAI SHREEYA **20241A1212**

K. TANMAI **20241A1247**

TABLE OF CONTENTS

LIST OF FIGURES:

ABSTRACT

Road accidents are increasing worldwide and are causing millions of deaths each year. They impose significant financial and economic expenses on society. Existing research has mostly studied road accident prediction as a classification problem, which aims to predict whether a traffic accident may happen in the future or not without exploring the underneath relationships between the complicated factors contributing to road accidents. Every day thousands of people are killed and injured on our roads. Men, women or children walking, biking or riding to school or work, playing in the streets or setting out on long trips can become victims of road accidents leaving behind shattered families and communities. According to the World Health Organisation (WHO), each year approximately 1.35 million annual road accidents happen seriously injuring between 20 to 50 million people worldwide. It is the eighth leading cause of global death and may become seventh by 2030 if the current trend continues.

To predict road accidents with different injury severity, this work has considered different ensembles of ML models, like Random Forest (RF), Adaptive Boosting (AdaBoost), Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (L-GBM), and Categorical Boosting Shapley Additive Explanations (SHAP), Local Interpretable Model-agnostic Explanation (LIME), and layer-wise relevance propagation (LRP) are widely used post hoc techniques. SHAP uses the Shapley value, which is the average marginal contribution of a feature value over all possible coalitions. SHAP guarantees accuracy and consistency in explainability. Moreover, SHAP has better visualisation features making it more popular when communicating with stakeholders from diverse backgrounds.

Keywords: AdaBoost, RandomForest, Shapley Additive Explanations (SHAP)
Domain: Machine Learning

1. INTRODUCTION

1.1 Introduction to the Project:

In this research, we investigate road accident prediction as a classification problem that can classify an accident's severity into four categories: fatal, serious, minor, and non-injury. Since more than two classes (or multiple classes) are involved in this road accident severity prediction research problem, it becomes a multiclass classification problem. This research has considered ensemble ML algorithms to analyze road accident datasets. An ensemble ML algorithm combines several base models to produce optimal performance and reduce the dispersion of prediction. To improve the acceptability of road accident prediction, it is essential to understand these factors and their influence on a model, which is the primary focus of this research. Thus the Shapley value is analyzed to understand the feature contributions toward the target variable. The findings were used to determine the underlying relationships of the contributing factors to the road accident. The performances of the different ensemble ML models were analyzed in this research in terms of accurately predicting road accident severity, understanding the precision of prediction, and calculating the F1-score and recall. The following subsections briefly describe all the ML methods explored in this paper.

The process typically involves the following key steps:

Data Collection: Gathering relevant data is fundamental to the success of any machine learning model. In the context of accident score prediction, this data may include information about past accidents, such as the location, time, weather conditions, road type, and other contributing factors.

Data Preprocessing: Once collected, the data needs to be cleaned and organized. This step involves handling missing values, removing outliers, and transforming data into a format suitable for machine learning algorithms.

Feature Selection: Identifying the most relevant features (variables) that contribute to the prediction is crucial. Features might include factors like road conditions, traffic density, and historical accident data.

Model Training: Various machine learning algorithms, such as decision trees, random forests, or neural networks, can be employed to train the model. During this phase, the algorithm learns from historical data and establishes patterns that can be used for predictions.

Validation and Evaluation: The trained model needs to be validated using separate datasets to ensure its generalization capabilities. Evaluation metrics such as accuracy, precision, recall, and F1 score are used to assess the model's performance.

Deployment: Once the model proves effective, it can be deployed in real-world scenarios. This may involve integrating it into existing systems, such as traffic management or navigation apps, to provide real-time accident risk assessments.

Continuous Improvement: Machine learning models benefit from ongoing refinement. As new data becomes available, the model can be updated to improve accuracy and adapt to changing conditions

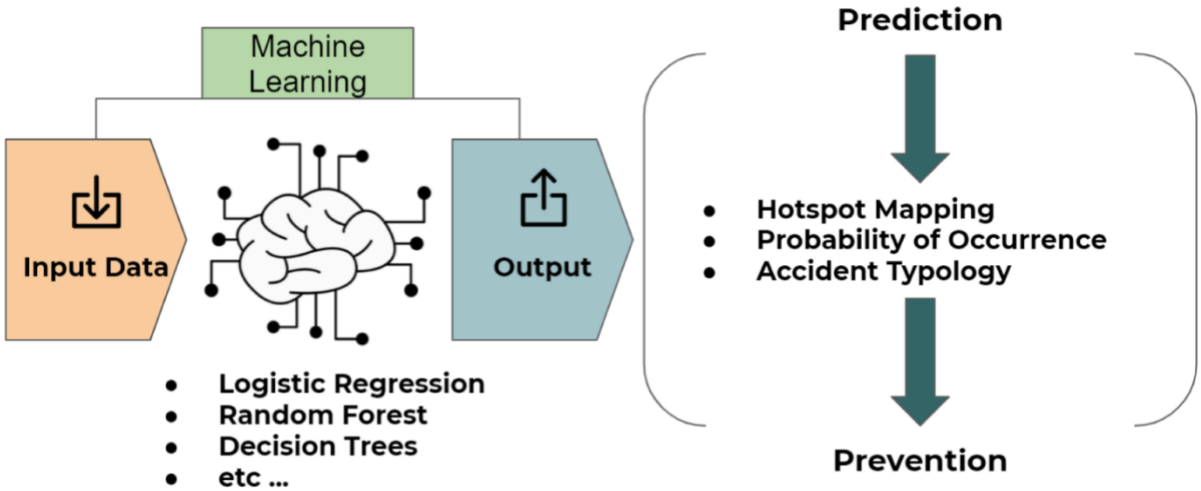


Fig 1: Accident Score Prediction

1.2 Existing Systems:

There are many different kinds of accident prediction system to choose from. This project aims to design a system that is an improvement from these current systems. Below are some methods which show how our system is different from other existing systems.

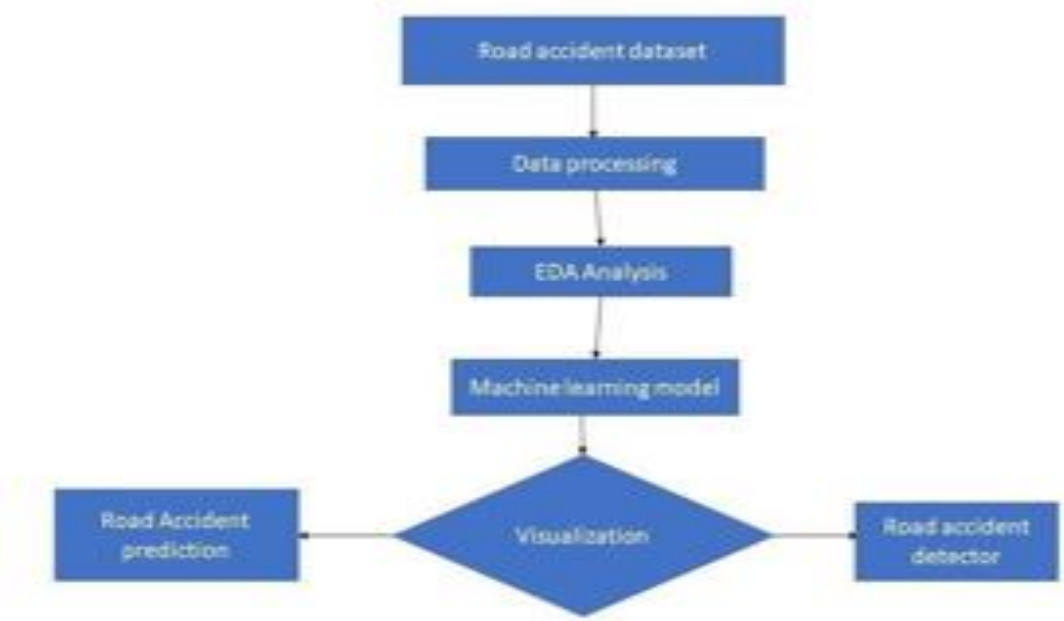


Fig 2: Existing System

Road Accident Prediction Model Using Machine Learning:

The model would be built utilising accident data records, which may aid in understanding the characteristics of a variety of elements such as driver behaviour, highway conditions, lighting conditions, and weather conditions, among others. This can assist users in calculating safety measures that are beneficial in preventing accidents. The model may be used to discover statistically significant characteristics that can be used to forecast the likelihood of collisions and injuries, as well as risk factors that can be utilised to minimise risk.

Road Accident Predictor:

Road Accident are a major cause of death worldwide leading to around 10.25 lakh deaths and 5 crores injuries ever year. Road accidents are extremely common. If you live in a sprawling metropolis like we do, chances are that we have heard about, witnessed, or even involved in one. Therefore, a system that can predict the occurrence of traffic accidents or accident-prone areas can potentially save lives. Although difficult, traffic accident prediction is not impossible. Accidents do not arise in a purely stochastic manner; their occurrence is influenced by a multitude of factors such as driver's physical conditions, car types, driving speed, traffic condition, road structure and weather. Paper is a deep learning python dynamic Routine maker which is designed to give the user a better understanding with the next day Traffic and help user to fulfill his/her sleep. Our mode Consider the following inputs speed, traffic condition, crash counts, road structure and weather to less obvious factors such as national holidays, the moon cycle and selective attention. Fortunately, several of such accident records are publicly available.

Various municipal and national government in the UK have made available rich datasets of Road Traffic Accidents (RTA) and their associated factors. By exploring this government datasets and external data sources, we aim to discover patterns that predict with high accuracy tells road accident to happens.

Advantages:

- A crash simulation produces results without actual destructive testing of a new car model.
- Tests can be performed quickly and inexpensively in a computer, which permits optimization of the design before a real prototype of the car has been manufactured.
- Stationed vehicle barriers on your property reduce the amount of traffic and speeding. Additionally, crash-rated barriers keep pedestrians or road workers safer.

Disadvantages:

- Accidents in the workplace can have a huge impact on any business, as they can cause loss of productivity, reduction of sales, low staff morale, loss of reputation – and at worst – closure. If any worker suffers an injury during their employment, they are entitled to workers' compensation.

1.3 Proposed System:

Despite advances in transportation safety measures, the increasing complexity of urban environments and the growing volume of vehicles on roadways pose ongoing challenges in effectively preventing accidents. Current accident prevention strategies often rely on historical data and static parameters, lacking the ability to dynamically assess real-time risk factors. There is a critical need for a robust accident risk score prediction system that integrates real-time data on weather conditions, road infrastructure, vehicle conditions, and driver behavior to proactively identify and quantify the risk of accidents. This predictive model should enable authorities to allocate resources strategically, implement targeted interventions, and improve overall road safety by reducing the frequency and severity of accidents. Addressing this gap in predictive capabilities is essential for creating a safer and more efficient transportation system, ultimately saving lives and minimizing the societal and economic impact of road accidents.

The overarching objective is to significantly reduce the frequency and severity of accidents, thereby minimizing the loss of life, injuries, and the associated societal and economic burdens. By addressing this critical gap in predictive capabilities, we aim to pave the way for a safer and more resilient transportation ecosystem.

2.REQUIREMENT ENGINEERING

2.1 Hardware Requirements:

- Laptop/PC

2.2 Software Requirements:

- Python programming Language
- Integrated Development Environment (IDE) : Jupyter Notebook or Anaconda
- Operating System
- Machine Learning Frameworks
- Libraries: Scikit-learn, XGBoost, LightGBM, CatBoost, Pandas, NumPy
- Algorithms: XGBclassifier, XGB regressor, LGBM classifier and regressor , CAT classifier and regressor , Random Forest regressor , Decision tree regressor, linear model, kfold.
- Data Visualization: Matplotlib, Seaborn, plot
- Model Interpretability: SHAP library

Python programming Language:

- Python has several implications across the software, web development, data science, and automation environments:
- Software testing and prototyping: Python can aid in software development tasks like build control, bug tracking, and testing.
- Machine learning and AI: Python is the number one language for machine learning and data science projects.
- Web development: Python and a framework called django can be used to build websites.
- Automation: Python can be used to automate repetitive tasks.

Integrated Development Environment (IDE) : Jupyter Notebook

- JupyterLab is the latest web-based interactive development environment for notebooks, code, and data.
- Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

Operating System:

- An operating system (OS) is a software that acts as an interface between a computer user and computer hardware. It's a collection of software that manages computer hardware resources and provides common services for computer programs.
- An operating system performs basic tasks such as:
  - File management
  - Memory management
  - Process management
  - Handling input and output
  - Controlling peripheral devices such as disk drives and printers
  - Recognizing input from the keyboard
  - Keeping track of files and directories on the disk

Libraries:

Scikit-learn:

- Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib

- Scikit-learn is a Python package designed to facilitate use of machine learning and AI algorithms. This package includes algorithms used for classification, regression and clustering such as random forests and gradient boosting.

XGBoost:

- XGBoost is a scalable and highly accurate implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms, being built largely for energizing machine learning model performance and computational speed.
- some features of XGBoost that make it so interesting: Regularization: XGBoost has an option to penalize complex models through both L1 and L2 regularization. Regularization helps in preventing overfitting. Handling sparse data: Missing values or data processing steps like one-hot encoding make data sparse.

Data Visualization:

Matplotlib:

- Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots

into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Create publication quality plots.

Seaborn:

- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data.

3.LITERATURE SURVEY

Prediction of Road Accidents Using Machine Learning Algorithms by R. Vanitha focuses on addressing the critical issue of road safety by introducing a Machine Learning-based Road Accident Prediction System. With an extensive dataset encompassing 1.6 million accident records from 2005 to 2015, the study employs Logistic Regression, Decision Tree, and Random Forest classification algorithms to predict accident severity. The main objectives include analyzing the interconnected factors influencing accidents, such as weather conditions, road surface, and vehicle status, and subsequently developing an accurate prediction model. Through rigorous data preprocessing techniques, including cleaning, visualization, and normalization, the study identifies key features affecting accidents. The experimental results underscore the efficacy of Random Forest in predicting accident severity and highlight significant contributing factors. The study emphasizes the importance of comprehending these interrelated aspects for effective transportation system management and proposes the potential integration of the model with platforms like Google Maps for real-time tracking and a user-police interaction web app. This research contributes essential insights to inform the development of strategies aimed at mitigating the frequency and severity of traffic accidents.

A study on road accident prediction and contributing factors using explainable machine learning models: analysis and performance, addresses the critical issue of road accidents and their severity, emphasizing the need for a comprehensive understanding of contributing factors. Focused on the New Zealand road accident dataset from 2016 to 2020, the research employs ensemble machine learning (ML) models, including Random Forest (RF), Decision Jungle (DJ), AdaBoost, Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (L-GBM), and Categorical Boosting (CatBoost), to predict accident severity across four categories: fatal, serious, minor, and non-injury. RF emerges as the most accurate classifier, achieving 81.45% accuracy. The study goes beyond traditional performance metrics, incorporating explainable ML techniques, particularly Shapley value analysis, to unveil the relationships between features and severity prediction. The global and local Shapley analyses reveal the significance of factors like road category and the number of vehicles involved. Retraining models with high-ranked features enhances performance, showcasing a 6%, 5%, and 8% improvement for DJ, AdaBoost, and CatBoost, respectively. The findings offer valuable insights for road safety professionals, policy design, and societal cost reduction associated with road accidents. Limitations include dataset specificity and the need for future exploration of advanced deep learning models to address imbalanced data.

The paper called Road Car Accident Prediction Using a Machine-Learning-Enabled Data Analysis, addresses the global issue of traffic accidents, emphasizing the need for effective data analysis and predictive models to reduce accidents and their associated fatalities. The study leverages machine learning and data analysis techniques to explore patterns in road car accident data, specifically focusing on accident severity, casualty count, and the number of vehicles involved. Utilizing a comprehensive dataset from the UK spanning 2005-2014, the authors employ four machine learning methods (decision trees, random forest classification, multinomial logistic regression, and naïve Bayes) to develop predictive models. The research highlights the significance of preprocessing, including feature removal, attribute generalization, and outlier removal, to enhance the quality of the dataset. Results demonstrate high accuracy in predicting accident-related variables, except for naïve Bayes. The findings contribute to a data-driven understanding of factors influencing road car accidents, providing insights for accident prevention strategies and the development of community-friendly cities. The methodology involves data preprocessing, cleaning, and transformation, with a focus on key variables such as weather conditions, light conditions, road type, and geographic factors.

The study done by Swati Mane focuses on leveraging machine learning, specifically association rule mining, to analyze and understand the influencing variables of road traffic accidents with the ultimate goal of reducing their occurrence. The research explores the decline in the number of accidents over the past 20 years and aims to break down the relationships between various factors contributing to accidents. The proposed methodology involves the development of an automatic modeling technique based on association rules to extract strong rules efficiently. The study emphasizes the importance of exploring accident-causing elements, such as road conditions, traffic flow, driver characteristics, and the environment, to enhance traffic safety. The methodology includes building a model using accident data records and utilizing machine learning algorithms for model training, evaluation, and testing. The advantages of the proposed approach include overcoming the accident rate, accurate prediction of accident severity, improving traffic safety management, and providing beneficial suggestions for road safety improvements. The project highlights the potential of using machine learning classification techniques to predict accident severity, offering insights into variables influencing accidents and proposing measures to enhance road safety.

4.TECHNOLOGY

4.1 A BRIEF HISTORY OF ML:

The history of Machine Learning (ML) traces back to the mid-20th century, with its roots in the field of artificial intelligence. The concept of machines that can learn from data has evolved significantly over the years, marked by key milestones and breakthroughs. Here is a highly condensed timeline of the development of ML and some of the significant occasions that have influenced its current state:

1950s: The foundation of ML was laid in the 1950s, with pioneers like Alan Turing proposing the idea of machines capable of learning from experience. Turing's work provided the theoretical basis for machine learning.

1960s-1970s: Early ML algorithms, such as the "Nearest Neighbor" algorithm, emerged during this period. Additionally, the development of the "Perceptron" by Frank Rosenblatt laid the groundwork for neural networks.

1980s: ML faced challenges and skepticism during the "AI Winter." Despite setbacks, researchers continued to explore various approaches, contributing to the foundation of symbolic reasoning systems and expert systems.

1990s: The resurgence of interest in ML occurred in the 1990s. The introduction of support vector machines, decision trees, and the development of algorithms like C4.5 marked significant progress.

1997: IBM's Deep Blue defeated world chess champion Garry Kasparov, showcasing the power of machine learning in strategic decision-making.

2000s: The rise of the internet and the availability of vast amounts of data fueled the development of data-driven ML techniques. Support vector machines, random forests, and ensemble methods gained popularity.

2010s: Deep Learning emerged as a dominant force in ML, with neural networks experiencing a renaissance. Breakthroughs in image and speech recognition, as well as natural language processing, highlighted the potential of deep learning models.

2012: The ImageNet competition marked a turning point with the victory of AlexNet, a deep convolutional neural network, significantly advancing the field of computer vision.

2015: The introduction of Generative Adversarial Networks (GANs) by Ian Goodfellow opened up new possibilities in generative modeling.

2018: Transformer models, such as BERT and GPT, revolutionized natural language processing tasks, achieving state-of-the-art performance.

2020s: Ongoing developments focus on addressing ethical concerns, interpretability, and robustness of ML models, shaping the future of responsible AI.

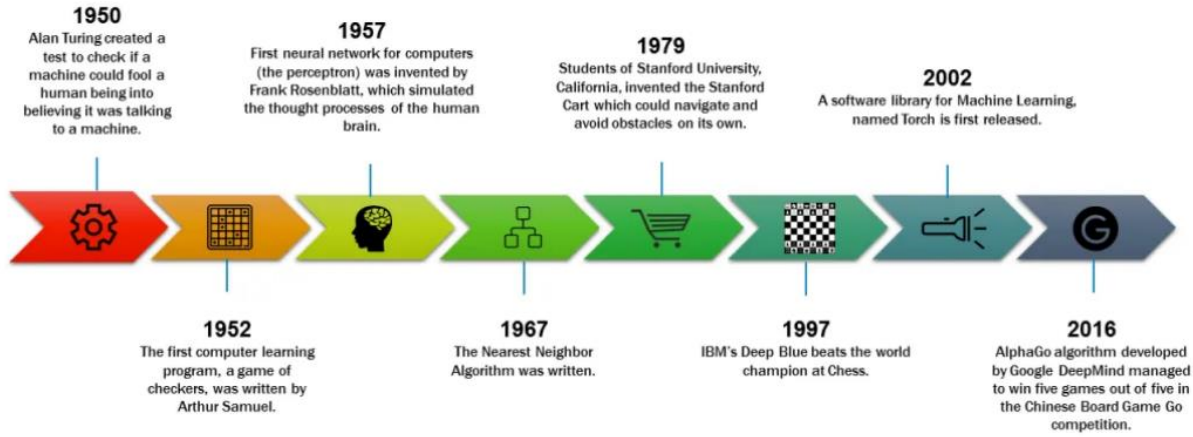


Fig 3: History of ML

4.2 THE EMERGENCE OF ML:

The emergence of Machine Learning (ML) marks a watershed moment in the annals of technology and artificial intelligence, signifying a convergence of theoretical foundations and technological advancements. While the nascent ideas of ML were conceived in the mid-20th century by visionaries like Alan Turing, it wasn't until the late 20th century and the early 21st century that ML truly came into its own. The catalyst for this resurgence was a confluence of factors: increased computational power, the proliferation of digital data, and advancements in

algorithmic design. These developments created an environment conducive to the exploration of sophisticated learning models capable of discerning intricate patterns within massive datasets.

The pivotal breakthrough came with the rise of neural networks, particularly the advent of deep learning architectures. These neural networks, inspired by the human brain's structure, demonstrated an unprecedented capacity to autonomously learn and adapt. Deep learning algorithms, with their ability to automatically extract hierarchical features from data, transformed ML from a conceptual framework into a practical tool with broad-reaching applications.

4.3 INTRODUCTION TO ML:

Machine Learning (ML) stands at the forefront of artificial intelligence (AI), representing a paradigm shift in computational approaches. Unlike traditional programming, where explicit instructions dictate a computer's behavior, ML empowers machines to learn from data and improve their performance autonomously. At its core, ML enables computers to recognize patterns, derive insights, and make predictions without explicit programming. The key distinction lies in the ability of ML algorithms to generalize knowledge from specific examples, allowing them to handle new, unseen data.

The learning process in ML involves exposing algorithms to labeled training data, where the algorithm learns patterns and relationships. This learned knowledge is then applied to new data to make predictions or solve problems. ML encompasses various types, including supervised learning, where the algorithm is trained on labeled data, unsupervised learning, focusing on discovering patterns in unlabeled data, and reinforcement learning, which involves decision-making based on interactions with an environment and feedback mechanisms.

The applications of ML are vast and impactful, ranging from image and speech recognition to natural language processing and recommendation systems. ML algorithms power virtual assistants, fraud detection systems, and personalized content recommendations, contributing to advancements across diverse industries such as healthcare, finance, and manufacturing.

As ML advances, innovative techniques like deep learning, a subset of ML involving neural networks with multiple layers, are gaining prominence. Deep learning enables machines to comprehend intricate patterns within data, leading to breakthroughs in tasks like image classification and language translation.

ML is revolutionizing problem-solving and decision-making, offering machines the capability to adapt and enhance their performance over time. The continuous evolution of ML promises to

reshape industries, improve efficiency, and contribute to the development of intelligent systems capable of navigating the complexities of the modern technological landscape.

4.4 ML WORKING PRINCIPLE:

The working principle of machine learning (ML) revolves around the idea of enabling computers to learn from data and improve their performance over time without being explicitly programmed. The key steps involved in the working of machine learning are as follows:

1.Data Collection:

The first step in any machine learning project is to gather relevant data. The quality and quantity of data play a crucial role in the effectiveness of the learning process.

2.Data Preprocessing:

Once the data is collected, it needs to be prepared for the learning algorithms. This involves cleaning the data, handling missing values, and transforming it into a format suitable for analysis.

3.Feature Selection and Engineering:

ML models rely on features (attributes) to make predictions. Feature selection involves choosing the most relevant features, while feature engineering may involve creating new features that enhance the model's performance.

4.Model Selection:

Choosing the appropriate machine learning algorithm depends on the nature of the problem (classification, regression, clustering) and the characteristics of the data. Common algorithms include decision trees, support vector machines, neural networks, and others.

5.Training the Model:

In the training phase, the selected model is fed with the labeled training data. The algorithm learns the patterns and relationships within the data to make predictions.

6.Evaluation:

The trained model is then evaluated using a separate set of data (testing data) that it has not seen before. This step assesses the model's performance and generalization capabilities.

7.Fine-Tuning:

Based on the evaluation results, the model may be fine-tuned by adjusting parameters or

selecting different algorithms to improve its accuracy and effectiveness.

8.Prediction or Inference:

Once the model is trained and fine-tuned, it can be deployed to make predictions or classifications on new, unseen data. This is the inference phase where the model applies its learned knowledge.

9.Feedback Loop:

In some cases, the model may have a feedback loop where it continuously learns from new data, adapting and improving its predictions over time. This iterative process enhances the model's performance as it encounters more diverse datasets.

Deployment:

Lastly, the trained and validated model is deployed for real-world use, making predictions on new data and potentially influencing decision-making processes.

Machine learning is a dynamic field, and the effectiveness of the models depends not only on the algorithms but also on the quality of data, feature engineering, and continuous monitoring and improvement. The iterative nature of the ML process allows models to adapt to changing conditions and data patterns, making them valuable tools in various applications.

4.5 VARIOUS APPLICATIONS OF ML:

- Healthcare
- Finance
- E-commerce
- Autonomous Vehicles
- Natural Language Processing (NLP)
- Image and Video Analysis
- Manufacturing and Industry
- Cybersecurity

Healthcare:

Machine learning has emerged as a groundbreaking technology with transformative applications in healthcare, revolutionizing the industry in multiple ways. One key application is disease diagnosis and prediction, where ML models analyze vast amounts of patient data, including medical records and imaging, to assist in early and accurate identification of diseases. Personalized treatment plans, another vital aspect, leverage ML algorithms to tailor

interventions based on individual patient characteristics, optimizing therapeutic outcomes. ML contributes significantly to drug discovery and development by predicting potential drug candidates and streamlining the research process. In radiology and imaging analysis, ML enhances diagnostic accuracy by automating the interpretation of medical images, aiding healthcare professionals in detecting abnormalities. Clinical decision support systems utilize ML to provide real-time recommendations for treatment planning and medication selection. Genomic analysis benefits from ML in identifying genetic patterns and markers, guiding precision medicine approaches. Additionally, ML plays a crucial role in fraud detection within healthcare billing, ensuring financial integrity. Telemedicine and remote patient monitoring leverage ML for real-time analysis of data from wearable devices, facilitating virtual consultations and timely interventions. Epidemiological forecasting and natural language processing applications further highlight the diverse and impactful role of ML in improving healthcare delivery, patient outcomes, and medical research.

Fig 4: Healthcare in ML

Finance:

Machine learning has transformed the financial sector with a range of applications that enhance decision-making, risk management, and customer experiences. In credit scoring, ML algorithms analyze vast datasets to assess an individual's creditworthiness, enabling more accurate and personalized lending decisions. Fraud detection is another critical application where ML identifies patterns indicative of fraudulent activities, helping financial institutions safeguard against unauthorized transactions. Algorithmic trading utilizes ML models to analyze market

trends, predict stock movements, and execute trades at optimal times. Customer relationship management benefits from ML-driven insights, enabling personalized services, targeted marketing, and improved customer satisfaction. Risk management is enhanced through predictive analytics, allowing financial institutions to proactively identify and mitigate the potential risks.

Machine Learning benefits in Finance:

- improved accuracy;
- efficient fraud detection;
- algorithmic trading efficiency;
- personalized customer experiences;
- proactive risk management.

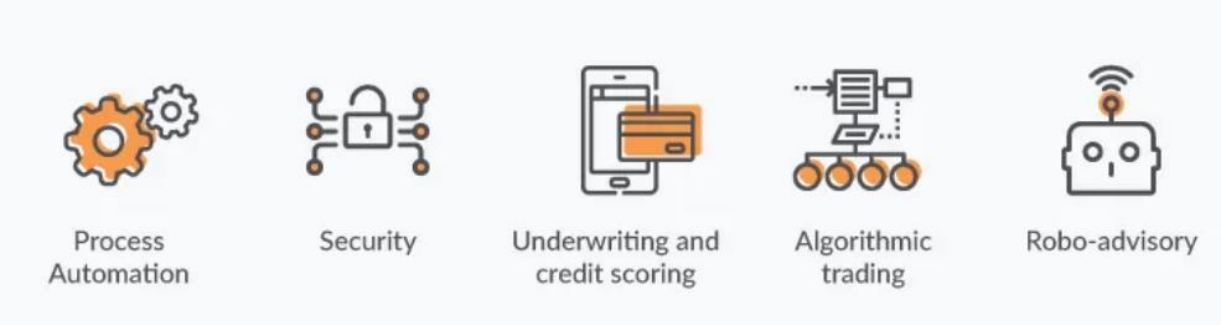


Fig 5: Finance in ML

E-commerce:

Machine learning (ML) has become a driving force in the e-commerce industry, revolutionizing various aspects of online retail. One prominent application is personalized product recommendations, where ML algorithms analyze customer behavior and preferences to suggest products tailored to individual tastes. Customer segmentation is another key area, allowing businesses to target specific demographics with personalized marketing campaigns. Price optimization, enabled by ML, involves dynamic pricing strategies based on real-time market conditions, demand, and competitor pricing. Fraud detection is crucial in e-commerce, and ML algorithms excel in identifying irregular patterns and preventing unauthorized transactions. Chatbots and virtual assistants, powered by natural language processing (NLP) algorithms, enhance customer interactions by providing real-time assistance and information.

ML benefits in -commerce:

- enhanced customer experience;

- dynamic pricing strategies;
- fraud prevention;
- efficient customer service;
- inventory management;
- streamlined marketing campaigns;
- recommendation engines;
- adaptive website design.

Fig 6: E-commerce

Autonomous Vehicles:

Machine learning plays a pivotal role in the development and deployment of autonomous vehicles, transforming the transportation landscape. Object detection and recognition are fundamental applications, where ML algorithms analyze sensor data to identify and classify objects in the vehicle's surroundings. Path planning utilizes ML to navigate through complex environments, making real-time decisions based on dynamic road conditions. Predictive maintenance ensures the health of vehicle components by analyzing data from sensors, reducing the risk of unexpected failures. ML contributes to advanced driver assistance systems (ADAS), enhancing safety features such as lane-keeping, collision avoidance, and adaptive cruise control. Continuous learning algorithms allow autonomous vehicles to adapt to evolving road scenarios,

improving overall performance and safety.

Benefits are:

- efficient path planning;
- predictive maintenance;
- adaptability;
- reduced environmental impact;
- improved traffic flow;
- enhanced accessibility.

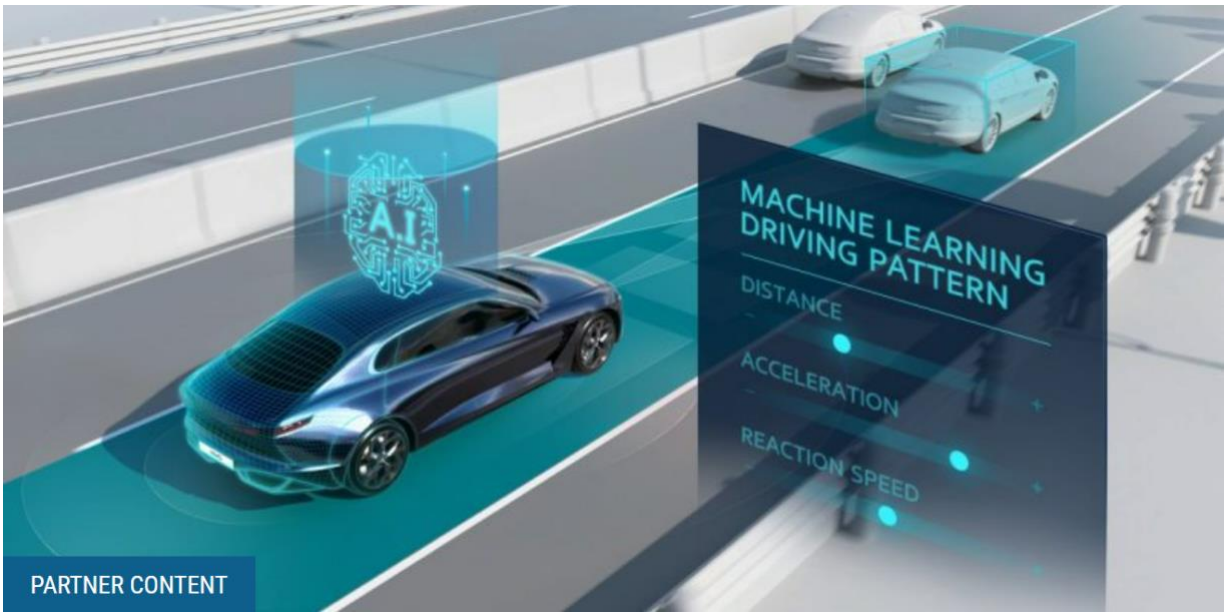


Fig 7: Autonomous Vehicles

Natural Language Processing:

Machine learning plays a crucial role in advancing Natural Language Processing, enabling computers to understand, interpret, and generate human-like language. Sentiment analysis, a common NLP application, leverages ML algorithms to analyze and understand opinions expressed in textual data, providing insights into user sentiments. Named Entity Recognition (NER) is another application, where ML models identify and classify entities such as names, locations, and organizations within text. Machine translation, facilitated by ML, enhances language translation capabilities, breaking down language barriers. Chatbots and virtual assistants utilize ML algorithms for language understanding, enabling human-like interactions and responses. Text summarization, speech recognition, and question-answering systems are additional NLP applications empowered by machine learning.

Machine learning benefits for NLP:

- improved sentiment analysis;
- efficient named entity recognition;
- accurate machine translation;
- automated text summarization;
- precise speech recognition.

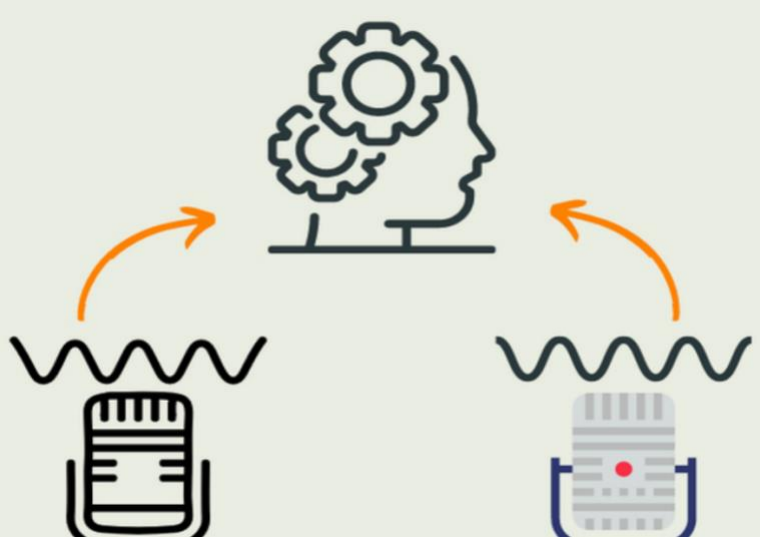


Fig 8: NLP

Image and Video Analysis:

Machine learning has ushered in a transformative era in image and video analysis, powering applications that range from facial recognition to autonomous vehicles. ML algorithms excel at identifying and classifying objects within images and videos, contributing to advancements in various fields. In healthcare, ML aids in medical imaging analysis, facilitating more accurate diagnoses through the interpretation of complex medical scans. In surveillance and security, facial recognition and object detection technologies enhance safety measures. The benefits of ML in image and video analysis include heightened accuracy in object recognition, increased automation in visual tasks, and substantial progress in domains such as healthcare, surveillance, and entertainment.

ML benefits for Image and Video Analysis:

- accurate object recognition;
- automation;
- medical diagnostics;
- enhanced security.

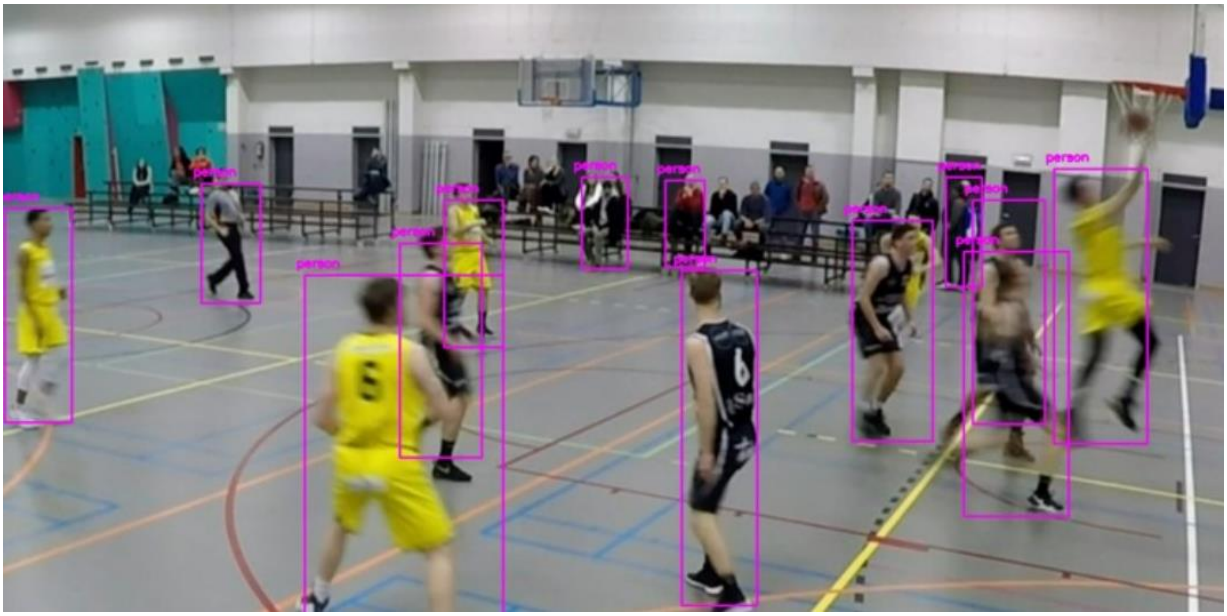


Fig 9: Sports video analysis using ML

Manufacturing and Industry:

In the realm of manufacturing and industry, machine learning applications drive optimization, efficiency improvement, and predictive maintenance. ML algorithms analyze sensor data to predict equipment failures, enabling proactive maintenance and minimizing downtime. Predictive analytics, another facet of ML, plays a pivotal role in optimizing production schedules and inventory management. Embracing the principles of Industry 4.0, ML-driven smart factories bring forth enhanced automation, flexibility, and overall productivity.

ML benefits for manufacturing industry:

- predictive maintenance;
- optimized production;
- inventory management;
- automation and flexibility.

Cybersecurity:

In the ever-evolving landscape of cybersecurity, machine learning emerges as a key player, fortifying defense mechanisms against advanced cyber threats. ML algorithms analyze intricate patterns in network traffic and user behavior to identify anomalies indicative of potential cyber threats. Intrusion detection systems, spam filters, and malware detection benefit from ML's adaptability, enabling them to evolve alongside emerging cyber threats.

ML benefits for Cybersecurity:

- advanced threat detection;
- real-time monitoring;
- adaptive security measures;
- improved incident response.

4.6 DRAWBACKS OF ML:

While machine learning (ML) has revolutionized various industries, it is not without its drawbacks. Understanding these limitations is crucial for harnessing the technology effectively and responsibly.

Data Dependency:

Machine learning models heavily rely on data for training and decision-making. The quality and quantity of the training data directly impact the model's performance. Biased or incomplete datasets can lead to skewed results, reinforcing existing prejudices and limiting the model's generalization to diverse scenarios.

Lack of Interpretability:

Many machine learning models, especially complex ones like deep neural networks, operate as "black boxes," making it challenging to interpret their decision-making processes. This lack of transparency raises concerns about accountability and the ability to understand and explain the rationale behind a model's predictions.

Overfitting and Underfitting:

ML models can face challenges in finding the right balance between overfitting and underfitting. Overfitting occurs when a model performs well on training data but fails to generalize to new, unseen data. On the other hand, underfitting happens when a model is too simplistic and cannot capture the underlying patterns in the data.

Ethical Concerns and Bias:

Biases present in training data can result in biased machine learning models. If historical data reflects societal biases, the ML model may perpetuate or even exacerbate those biases. Addressing ethical concerns, ensuring fairness, and mitigating bias in ML models is an ongoing challenge.

Computational Resource Requirements:

Training complex ML models, especially deep learning models, demands significant computational resources. This includes powerful hardware, specialized processors, and substantial memory. As a result, smaller organizations or individuals may face challenges in

implementing and maintaining high-performance computing infrastructure.

Lack of Common Sense:

While ML models excel at pattern recognition based on training data, they often lack common sense reasoning abilities. These models may make predictions that defy human intuition, highlighting the need for incorporating domain knowledge and contextual understanding alongside data-driven insights.

Security Concerns:

Machine learning models are susceptible to adversarial attacks, where intentional manipulations of input data can mislead the model's predictions. Safeguarding ML systems against security threats and ensuring their robustness is an ongoing area of research.

Continuous Need for Training:

ML models require continuous training to adapt to changing patterns in data. This necessitates ongoing maintenance, monitoring, and updates, making it resource-intensive and requiring a dedicated effort to keep models relevant and effective over time.

While machine learning brings immense capabilities, acknowledging and addressing these drawbacks is essential for responsible and ethical deployment of ML technologies. Researchers and practitioners continue to work towards developing solutions to mitigate these challenges and enhance the reliability and fairness of machine learning systems.

5.DESIGN REQUIREMENT ENGINEERING

5.1 INTRODUCTION:

The software engineering discipline uses the general-purpose visual modelling language known as UML or Unified Modelling Language. It is utilised for defining, visualising, creating, and documenting the main software system artefacts. It is useful for creating and characterising software systems, particularly ones that use the idea of object orientation. Software engineering explains hardware and software operations.

The UML was developed by Grady Booch, Ivar Jacobson, and James Rumbaugh at Rational Software between 1994 and 1995. The Object Management Group (OMG) officially approved it as a standard in 1997.
The Object Management Group (OMG), an association of companies, oversees the open standard UML.

5.1.1 ACTIVITY DIAGRAM:

Instead of displaying the implementation of the system, the UML activity diagram is meant to display the control flow. Models are used to simulate concurrent and sequential actions.

The activity diagram can be used to depict the flow of work from one action to the next. It highlighted the presence of flow and the order in which it occurs. The activity diagram has been built with a fork, join, etc. to deal with the various sorts of flows, which can be sequential, branching, or concurrent.

It is also known as an object-oriented flowchart. It contains activities that call for using a string of methods or actions to mimic the behavioural diagram. An activity diagram is made up number of nodes joined together to depict an event.

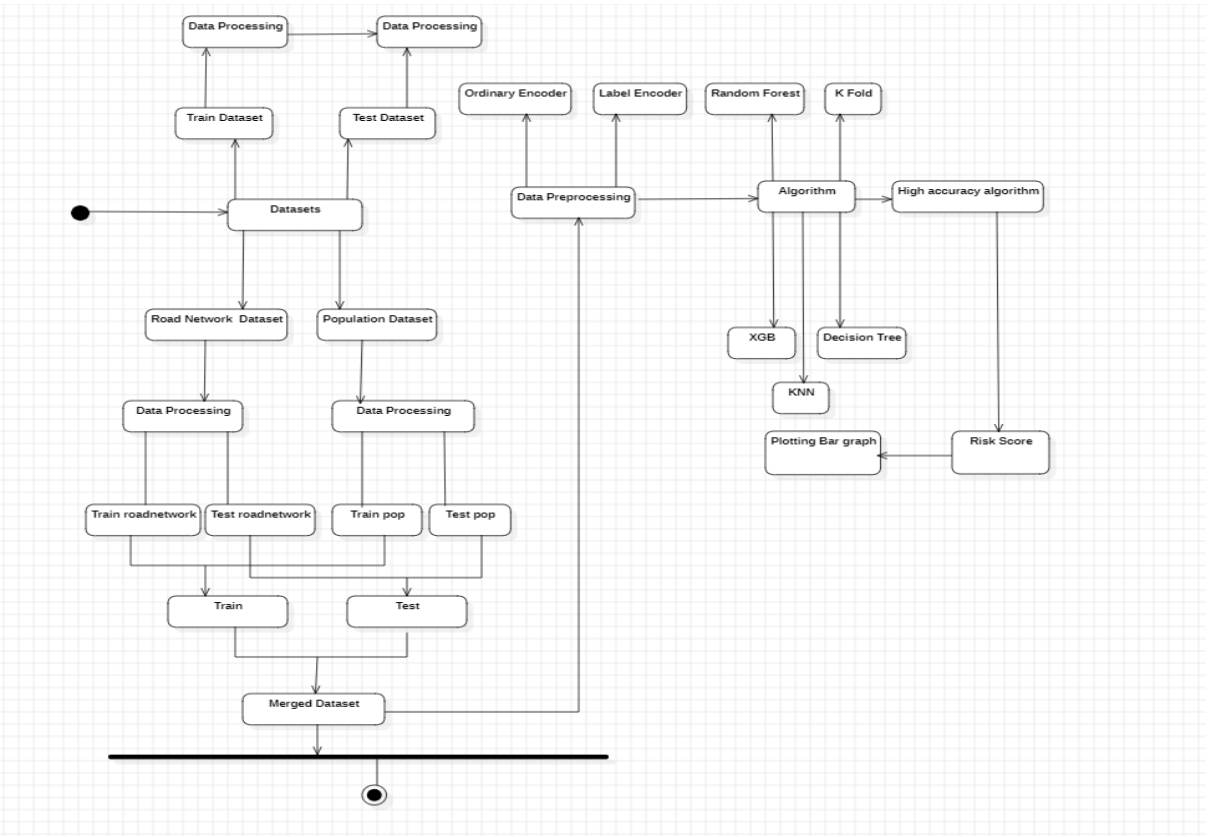


Fig 10: Activity diagram

5.1.2 SEQUENCE DIAGRAM:

The most popular type of interaction diagram is a sequence diagram. The interaction diagram is used to show how the system interacts with the environment. We use many forms of interaction diagrams to capture different elements and components of interaction in a system because visualizing interactions in a system can be difficult. A sequence diagram essentially shows how things interact in a sequence or in the order in which these interactions occur. A sequence diagram can also be called an event diagram or an event scenario. The sequence diagram shows the actions taken by the components of the system in the sequence. Businessmen and software engineers often use these diagrams.

This is used to simulate and illustrate the reason behind a complex function, process, or method. UML use case diagrams are also displayed using these case diagrams. Use to understand the precise operation of current or future systems. Imagine the flow of information between different elements or systems objects.

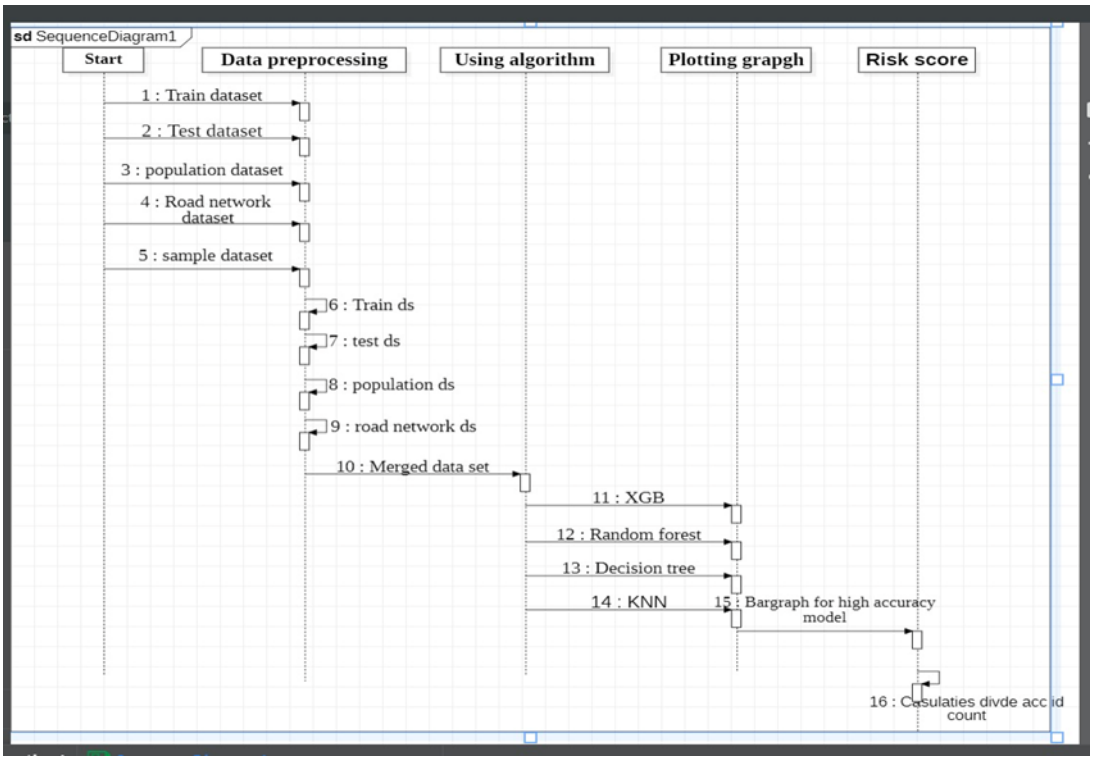


Fig 11: Sequence diagram

5.1.3 CLASS DIAGRAM:

A class diagram is a type of diagram in the Unified Modeling Language (UML) that represents the structure and relationships of a system's classes or objects. It provides a static view of the system, highlighting the classes, their attributes, methods, and the associations between them. Class diagrams are widely used in software development to visualize the design and structure of a system before its implementation.

Key elements of a class diagram include:

1. Class:
    - A class is a blueprint or template for creating objects. It represents a concept or entity in the system.
    - The class name is usually placed at the top of the diagram and is often written in bold.
    - Example: Car, Employee, Customer.
2. Attributes:
    - Attributes are the properties or characteristics of a class.
    - They are listed beneath the class name and typically include data types.

- Example: Car class may have attributes like make, model, and year.

3. Methods:
    - Methods represent the behavior or actions that an object of the class can perform.
    - They are listed beneath the attributes and often include parameters and return types.
    - Example: Car class may have methods like startEngine(), stopEngine(), and drive().
4. Associations:
    - Associations show the relationships between classes. They illustrate how one class is connected to another.
    - Multiplicity notations indicate the number of instances involved in the association.
    - Example: An association between Car and Driver classes could be represented to show that each Car is associated with one or more Drivers.
5. Inheritance:
    - Inheritance represents an "is-a" relationship between classes, where one class (subclass or child) inherits the properties and behaviors of another class (superclass or parent).
    - It is depicted using a solid line with an open arrowhead pointing from the subclass to the superclass.
6. Encapsulation:
    - Encapsulation is depicted by showing class attributes and methods as private (+), public (-), or protected (#).
    - Public members are accessible from outside the class, while private members are only accessible within the class.
7. Abstraction:
    - Abstraction involves showing only the essential features of a class and hiding unnecessary details.
    - It helps in simplifying the complex system by focusing on the relevant aspects.
8. Aggregation and Composition:
    - Aggregation and composition represent "has-a" relationships between classes.
    - Aggregation is a weaker relationship, while composition implies a stronger relationship, often indicating that one class is part of another.

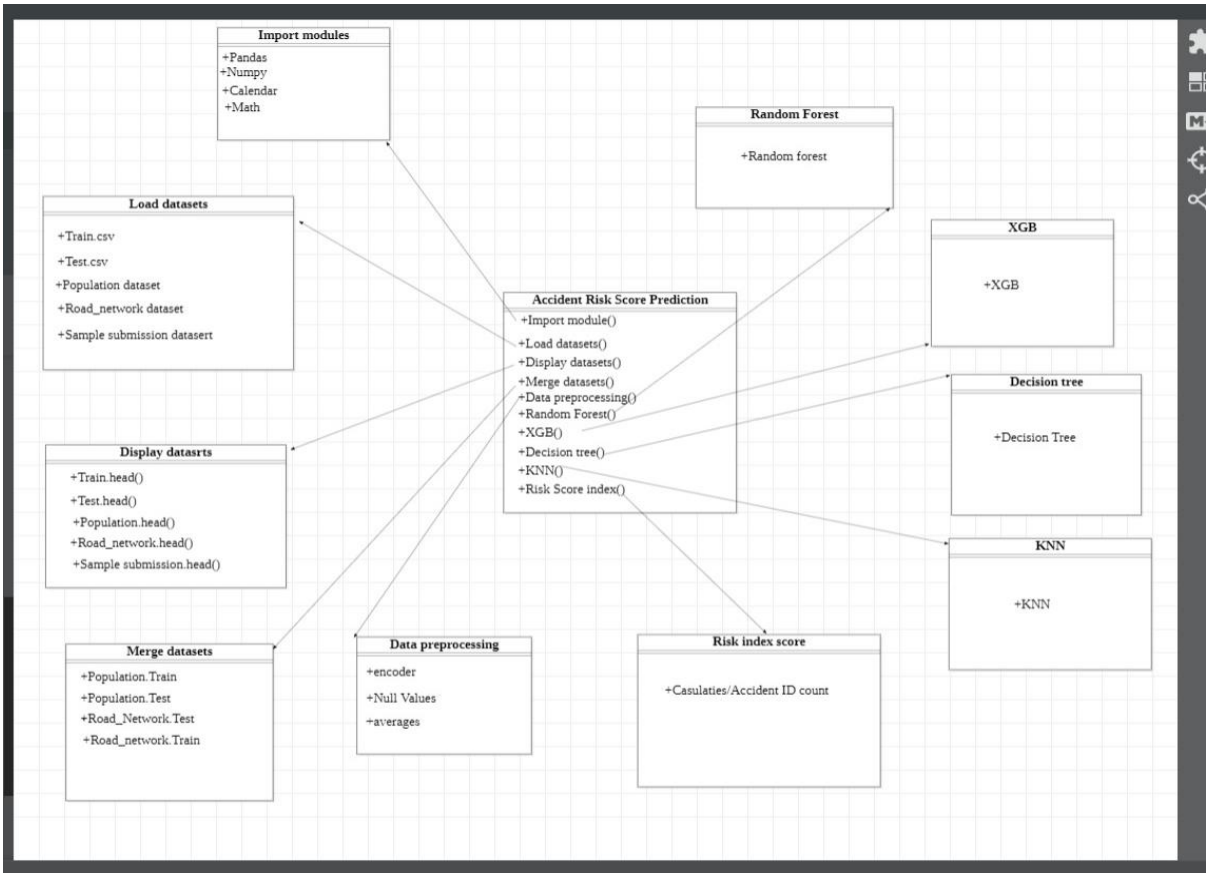


Fig 12: Class Diagram

5.1.4 USE CASE DIAGRAM:

A use case diagram is a type of diagram in the Unified Modeling Language (UML) that illustrates the interactions between actors and a system or a subsystem. It provides a high-level view of the system's functionality from the perspective of its users (actors) and how they interact with the system to accomplish specific goals or tasks.

Key elements of a use case diagram include:

1. **Actor:**
    - An actor is an external entity that interacts with the system. Actors can be people, other systems, or even physical entities.
    - Actors are represented by stick figures or any other appropriate symbols.
    - Example actors: "Customer," "Administrator," or "External System."
2. **Use Case:**
    - A use case represents a specific functionality or a set of related functionalities that the system provides to its users (actors).
    - Use cases are represented by ovals and are connected to the actors by lines to indicate the association between an actor and a use case.

- Example use cases: "Make a Purchase," "Login," or "Generate Report."

3. **Association:**
    - Associations are the lines connecting actors and use cases, indicating that an actor is involved in a particular use case.
    - Associations represent communication or interaction between actors and use cases.
4. **System Boundary:**
    - The system boundary is a box that encloses all the use cases of the system. It represents the scope or boundary of the system being modeled.
5. **Include Relationship:**
    - The include relationship is used to show that one use case includes the functionality of another use case.
    - It is represented by a dashed arrow from the including use case to the included use case.
6. **Extend Relationship:**
    - The extend relationship is used to indicate that a use case can be extended with additional behavior under certain conditions.
    - It is represented by a dashed arrow from the extended use case to the extension.
7. **Generalization (Inheritance):**
    - Generalization is used to represent relationships between use cases, similar to inheritance in object-oriented programming.
    - It is represented by a solid line with an open arrowhead pointing from the specialized (child) use case to the generalized (parent) use case.

Use case diagrams are valuable tools for system analysts and designers during the requirements gathering and analysis phase of software development. They help in visualizing the system's functionality, understanding user interactions, and identifying potential areas for system improvement. Use case diagrams also serve as a foundation for creating detailed use case descriptions and provide a basis for testing scenarios during the development process.

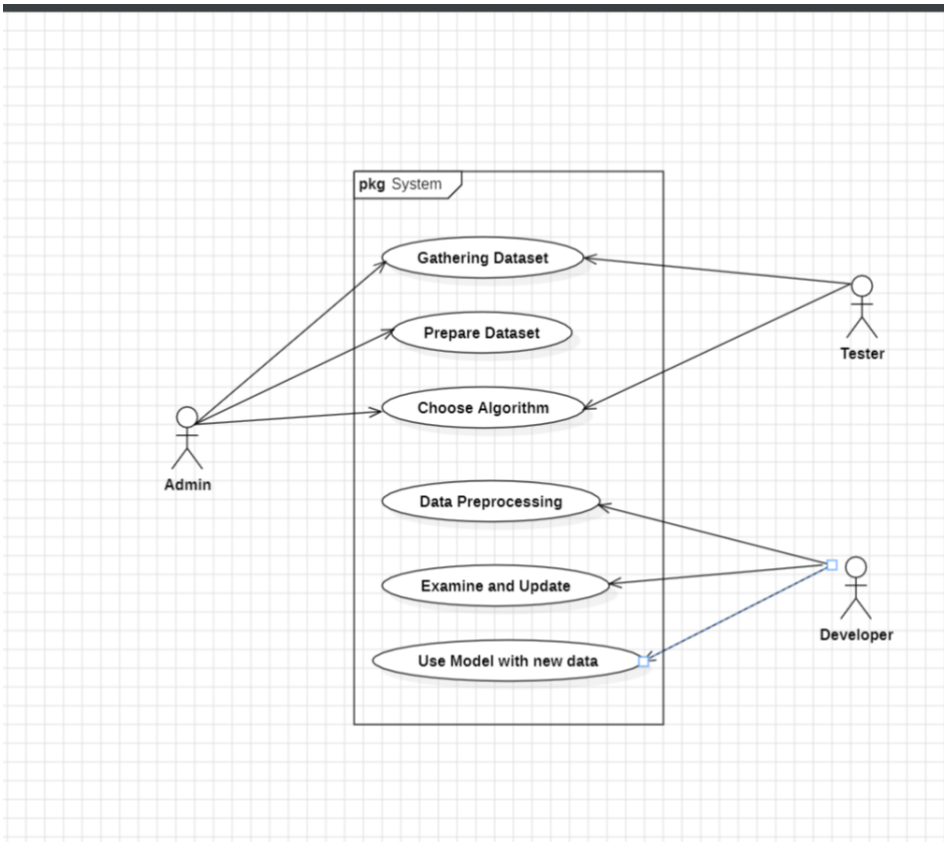


Fig 13: Use case diagram

5.1.5 ARCHITECTURE DIAGRAM:

Concepts, including principles, parts, and components, whose building structure is represented in a graphic diagram is an Architecture diagram.

A structure's architecture is its unified set of concepts. These ideas are frequently represented using four degrees of abstraction. Which are:

1. Conceptual Level: Outlining major concepts
2. Logical Level: Illustrating the principles (i.e., how the ideas operate) of one or more concepts by a logical design that includes at least their core components.
3. Physical Level: Displaying an element-based component design

The suppliers and products used to implement components are displayed at the implementation level. The diagrams of the logical levels of the architecture form each visualization of this page. An architectural diagram often includes a mixture of logical and physical aspects. An architectural diagram is a graphic representation of each component that forms a system,

whether fully or partially. Above all, it helps to understand the layout of a system or application by engineers, designers, stakeholders and everyone else.

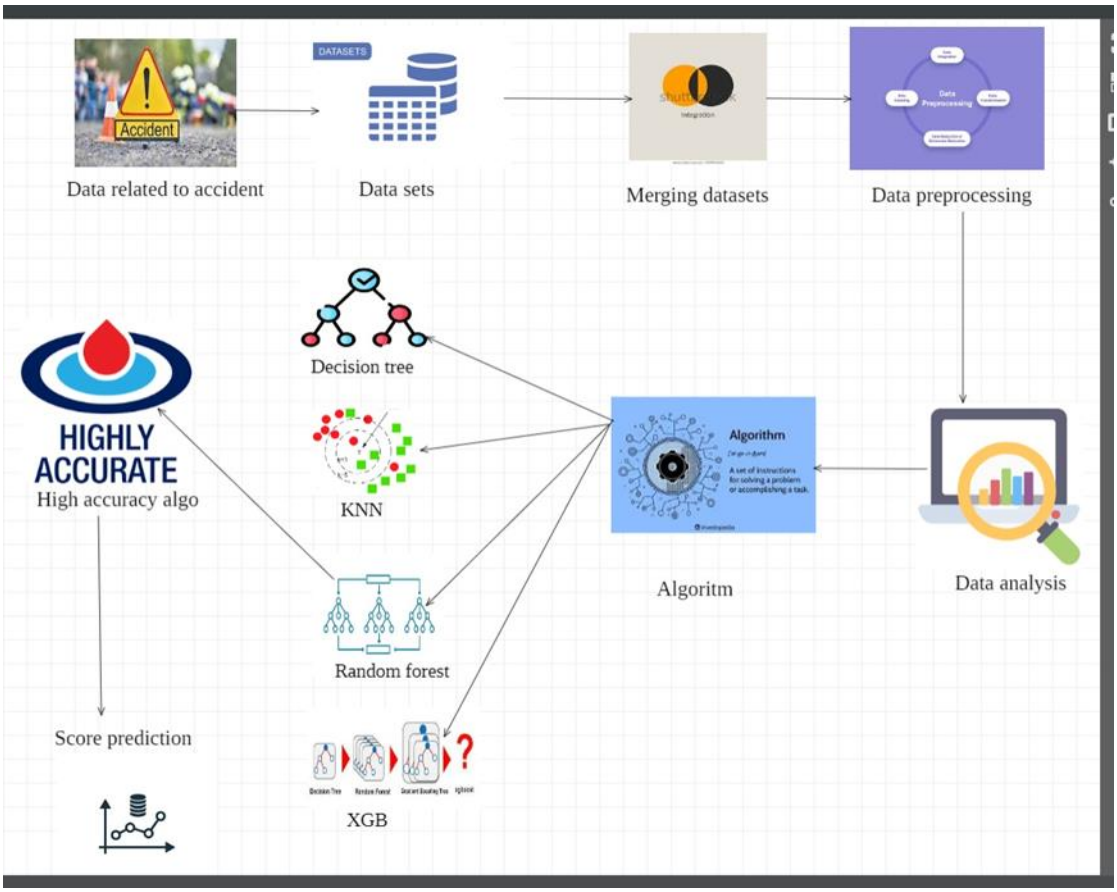


Fig 14: Architecture diagram

6.IMPLEMENTATION

6.1 Algorithms Used in the project:

1. Xgboost:

XGBoost (eXtreme Gradient Boosting) is an optimized and scalable machine learning library renowned for its speed and performance in solving regression, classification, and ranking problems. It's based on an ensemble learning technique known as gradient boosting, where multiple weak prediction models, typically decision trees, are combined to create a strong predictive model.

Key Features of XGBoost:

1. Gradient Boosting Algorithm:
    - XGBoost uses the gradient boosting framework that sequentially builds an ensemble of weak learners, focusing on the errors made by previous models. It minimizes a loss function by optimizing the gradients of the model.
2. Tree-Based Learning:
    - XGBoost primarily relies on decision trees as base learners. Each tree is built sequentially, and subsequent trees correct the errors of previous ones, making the ensemble more robust.
3. Regularization:
    - To prevent overfitting, XGBoost employs regularization techniques such as L1 (Lasso) and L2 (Ridge) regularization penalties on leaf weights and tree complexity. This helps in controlling model complexity and improving generalization.
4. Handling Missing Values:
    - XGBoost has in-built capabilities to handle missing values during the training phase. It automatically learns the best direction to take when missing values are encountered.
5. Parallel and Distributed Computing:
    - XGBoost is highly optimized for speed and scalability. It supports parallel and distributed computing, making it efficient and capable of handling large datasets.
6. Customization and Tuning:
    - Users can tune various hyperparameters in XGBoost to optimize model performance. Parameters related to tree construction, learning rates, and regularization can be adjusted based on the specific requirements of the problem.
7. Support for Multiple Languages:

- Originally implemented in C++, XGBoost also provides interfaces for various programming languages such as Python, R, Java, and Scala, making it accessible and usable across different platforms.

Advantages of XGBoost:

- High Performance: XGBoost is known for its speed and efficiency due to optimized algorithms and parallel computing techniques.
- Robustness: It handles different types of data and is robust against outliers and overfitting.
- Feature Importance: It provides insights into feature importance, aiding in feature selection and understanding model behavior.
- Flexibility: XGBoost can be used for both regression and classification problems across various domains and datasets.

2. LightGBM:

LightGBM (Light Gradient Boosting Machine) is an efficient, high-performance gradient boosting framework developed by Microsoft. It's designed for large datasets and provides fast training speeds while maintaining high accuracy. LightGBM belongs to the family of gradient boosting algorithms and shares similarities with other boosting techniques, such as XGBoost and CatBoost.

Key Features of LightGBM:

1. Gradient Boosting Algorithm:
    - LightGBM, like other boosting algorithms, works on the principle of creating an ensemble of weak learners (decision trees, in most cases) and sequentially improving predictions by focusing on the errors made by previous models.
2. Leaf-Wise Tree Growth:
    - LightGBM grows trees leaf-wise instead of level-wise, which typically results in faster training times. It chooses the leaf with the maximum delta loss to grow the tree, thus making the model more accurate with fewer leaves.
3. Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB):
    - GOSS reduces the number of data instances used for training by keeping instances with large gradients, significantly reducing the training time. EFB bundles exclusive features to reduce memory usage and speed up training.
4. Handling Categorical Features:
    - LightGBM efficiently handles categorical features by converting them into numerical values using techniques like the 'one-hot' encoding, optimizing memory usage without the need for extensive preprocessing.

5. Customizable Parameters:
    - LightGBM provides a wide range of hyperparameters to tune, allowing users to customize various aspects of the model, such as tree structure, learning rate, and regularization, to optimize performance.
6. Parallel and GPU Learning:
    - It supports parallel and GPU learning, leveraging multiple CPU cores and GPUs to speed up training and inference, making it scalable and suitable for large datasets.
7. Cross-Validation and Early Stopping:
    - LightGBM supports k-fold cross-validation for model evaluation and early stopping, which helps prevent overfitting by stopping training when the validation scores no longer improve.

Advantages of LightGBM:

- Speed and Efficiency: LightGBM's leaf-wise tree growth and optimizations make it extremely fast and memory-efficient, particularly for large datasets.
- Scalability: Its ability to handle large-scale data and use parallel computing makes it highly scalable.
- High Accuracy: Despite its speed, LightGBM maintains high predictive accuracy, making it suitable for various machine learning tasks.

3. CatBoost :

CatBoost is a high-performance, open-source gradient boosting library developed by Yandex, designed specifically to handle categorical features efficiently. It's an abbreviation of "Category Boosting." CatBoost is based on the gradient boosting framework and shares similarities with other boosting algorithms such as XGBoost and LightGBM but specializes in handling categorical variables seamlessly.

Key Features of CatBoost:

1. Categorical Feature Handling:
    - CatBoost automatically handles categorical features without requiring explicit encoding or preprocessing. It uses an efficient algorithm for dealing with categorical variables, which reduces the need for feature engineering.
2. Gradient Boosting Algorithm:
    - Similar to other boosting algorithms, CatBoost builds an ensemble of decision trees in a sequential manner, improving model performance by focusing on correcting errors made by previous models.
3. Optimized Handling of Numerical Data:

- CatBoost optimizes numerical data handling by applying various transformations and reducing memory usage, enhancing overall performance.

4. Robustness to Overfitting:
    - It includes built-in strategies to prevent overfitting, such as early stopping, regularization, and robust handling of outliers, ensuring better generalization to unseen data.
5. Support for Various Data Types:
    - CatBoost supports not only categorical and numerical features but also text data, making it versatile in handling different types of inputs.
6. Cross-Validation and Hyperparameter Tuning:
    - It provides built-in cross-validation support for model evaluation and automatic hyperparameter tuning, making it easier for users to optimize their models effectively.
7. Scalability and Speed:
    - CatBoost is designed to be scalable and performs well on large datasets. It leverages parallel processing and GPU support to improve training speed.
8. Feature Importance Analysis:
    - CatBoost offers insights into feature importance, helping users understand which features contribute the most to the model's predictions.

Advantages of CatBoost:

- Efficient Handling of Categorical Data: CatBoost's ability to handle categorical variables without extensive preprocessing makes it convenient and efficient.
- Strong Performance: It delivers competitive performance in terms of accuracy and generalization, especially in scenarios where categorical features play a significant role.
- Ease of Use: Its simplicity in handling categorical features reduces the complexity of the preprocessing pipeline, making it more user-friendly.

4. RandomForest Classifier:

The RandomForestClassifier is an ensemble learning method based on decision tree algorithms, known for its robustness and versatility in classification tasks. It belongs to the family of ensemble techniques that aggregate predictions from multiple individual models to make more accurate and stable predictions.

Key Features of RandomForestClassifier:

1. Ensemble of Decision Trees:
    - RandomForestClassifier builds multiple decision trees during training. Each tree is trained on a random subset of the dataset (bootstrapped samples) and a random

subset of features (feature bagging). This randomness helps in reducing overfitting and improving generalization.

2. Randomness and Diversity:
    - It introduces randomness in two ways: by sampling data points with replacement (bootstrap) and by considering only a subset of features at each split in the decision tree. This randomness creates diverse trees, leading to a more robust overall model.
3. Voting Mechanism for Classification:
    - In the case of classification tasks, RandomForestClassifier uses a voting mechanism to make predictions. Each tree in the forest predicts the class label, and the final prediction is determined by a majority vote among the individual trees.
4. Handling Nonlinear Relationships:
    - RandomForestClassifier is capable of capturing nonlinear relationships between features and the target variable, making it suitable for complex classification problems.
5. Robustness to Overfitting:
    - The randomness and aggregation of multiple trees help reduce the risk of overfitting, making RandomForestClassifier less sensitive to noise and outliers compared to individual decision trees.
6. Feature Importance:
    - RandomForestClassifier can provide a measure of feature importance, showing the contribution of each feature in making predictions, aiding in feature selection and understanding the data.

Advantages of RandomForestClassifier:

- Robust and Versatile: It performs well in various scenarios and is robust to noisy data and overfitting.
- Handles High Dimensionality: It can handle datasets with a large number of features without much feature engineering.
- Reduces Variance: Ensemble methods, like random forests, often lead to better generalization by reducing variance while maintaining low bias.

5. Decision Tree Regressor:

A Decision Tree Regressor is a machine learning algorithm used for solving regression problems. Unlike classification tasks that predict categorical labels, regression tasks involve predicting continuous numerical values. Decision trees, the fundamental building blocks of this model, create a tree-like structure by recursively partitioning the input space into regions and making predictions based on the average (or mean) value of the target variable within these regions.

Key Features of Decision Tree Regressor:

1. **Tree Structure:**
    - Decision trees consist of nodes representing features, branches representing decision rules, and leaf nodes representing the predicted output. Each internal node represents a decision based on a feature, splitting the data into smaller subsets.
2. **Recursive Partitioning:**
    - The tree-building process involves recursively splitting the dataset into subsets based on the feature that provides the best split, determined by a criterion like mean squared error or variance reduction.
3. **Handling Nonlinear Relationships:**
    - Decision trees can capture nonlinear relationships between input features and the target variable, making them suitable for modeling complex relationships in the data.
4. **Simple Interpretability:**
    - Decision trees are easy to visualize and interpret, making them useful for understanding how predictions are made based on feature splits.
5. **Prone to Overfitting:**
    - Decision trees are prone to overfitting, especially when they grow deep and complex. Pruning techniques or setting constraints on tree depth and minimum samples per leaf can help prevent overfitting.
6. **Ensemble Methods:**
    - Decision trees are often used as base learners in ensemble methods like Random Forests or Gradient Boosting Machines to improve overall predictive performance.

Advantages of Decision Tree Regressor:

- **Interpretability:** Easy to understand and interpret, making it a useful tool for explaining predictions.

6. Linear Model:

Linear models are a class of statistical and machine learning models that assume a linear relationship between the input features and the target variable. These models work on the principle of fitting a linear equation to the observed data to make predictions. They are widely used in various fields due to their simplicity, interpretability, and effectiveness in many scenarios.

Key Characteristics of Linear Models:

1. **Linear Relationship:**
    - Linear models assume that the relationship between the input variables and the target variable can be represented by a linear equation.
2. **Coefficients and Intercepts:**
    - Linear models estimate coefficients (weights) for each feature to minimize the difference between predicted and actual values. These coefficients indicate the impact of each feature on the target variable.
3. **Simple and Interpretable:**
    - Linear models are straightforward and easy to interpret. Coefficients show the direction and magnitude of the impact that each feature has on the target variable.
4. **Types of Linear Models:**
    - **Linear Regression:** Used for predicting a continuous target variable. It finds the best-fitting linear relationship between the input variables and the target.
    - **Logistic Regression:** Primarily used for binary classification problems. Despite its name, it is a classification algorithm that predicts the probability of an observation belonging to a particular class.
5. **Assumptions:**
    - Linear models assume that the relationship between the input features and the target variable is linear. They also assume the absence of multicollinearity (high correlation between predictors), homoscedasticity (constant variance of residuals), and independence of errors.

Advantages of Linear Models:

- **Interpretability:** Coefficients provide insight into the impact of each feature on the target variable.
- **Simplicity:** Easy to implement and understand, making them a good starting point for many modeling tasks.
- **Efficiency:** Fast training and prediction times, especially for large datasets.

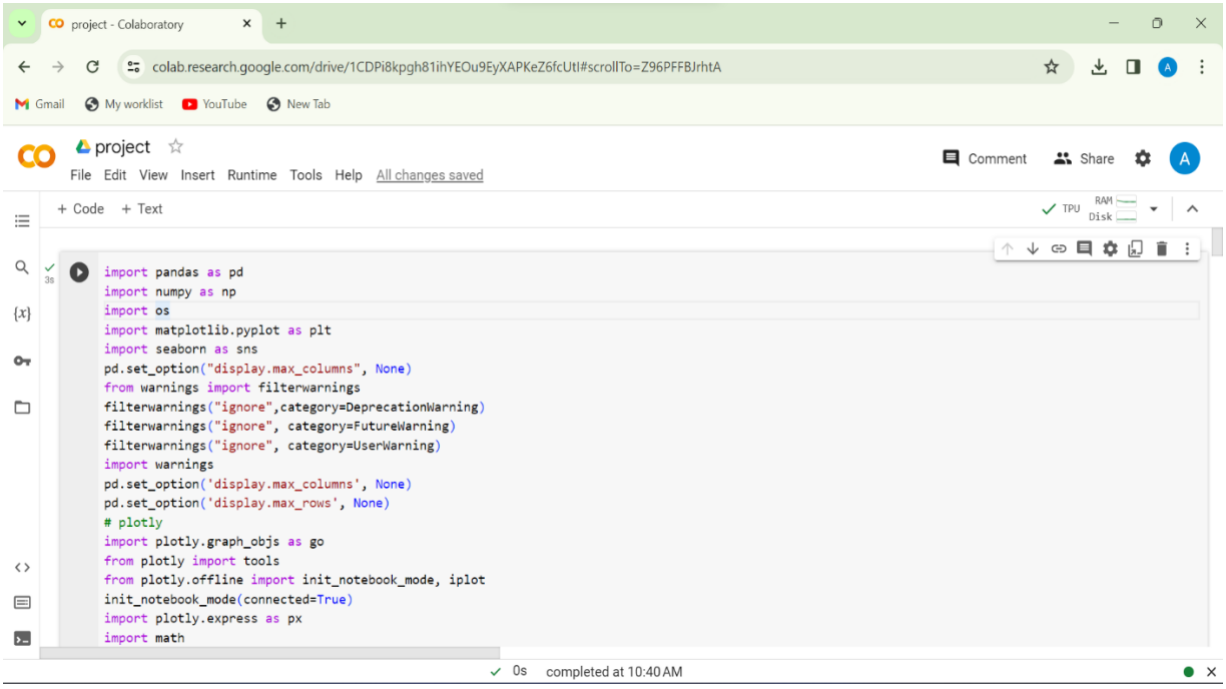
7. Kfold:

K-Fold Cross-Validation is a resampling technique used in machine learning for model evaluation and validation. It's an effective method to assess a model's performance, particularly when the dataset is limited or when there's a need to obtain a robust estimation of how well a model will generalize to new, unseen data.

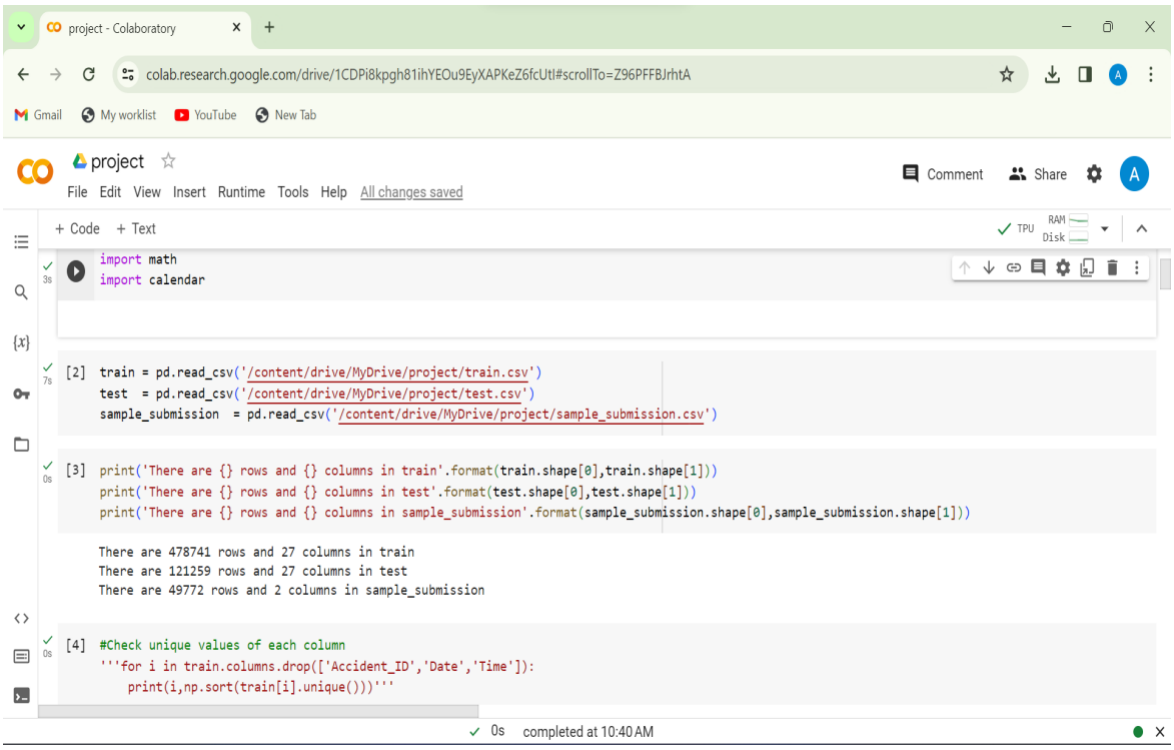
Key Components of K-Fold Cross-Validation:

1. Data Splitting:
    - The dataset is divided into k equal-sized folds or subsets. Typically, k is set to a value like 5 or 10, but it can vary based on the dataset size and specific requirements.
2. Training and Validation:
    - The model is trained and evaluated k times. In each iteration, one of the k subsets is used as the validation set, and the remaining k-1 subsets are used for training the model.
3. Evaluation Metric:
    - The performance metric (e.g., accuracy, mean squared error, etc.) is computed for each iteration using the validation set. The average performance across all k iterations is then calculated to obtain the final performance metric.
4. Benefits of K-Fold Cross-Validation:
    - Reduced Variance: K-Fold CV provides a more reliable estimate of model performance compared to a single train-test split by reducing the variance associated with a single split.
    - Utilizes Entire Dataset: It ensures that each data point is used for both training and validation, maximizing the use of available data for model evaluation.
    - Robustness: Helps in identifying models that perform consistently across different subsets of the data, indicating robustness in model performance.
5. Variants of Cross-Validation:
    - Stratified K-Fold: Ensures that each fold maintains the same class distribution as the original dataset, particularly useful for imbalanced datasets.
    - Repeated K-Fold: Repeats K-Fold CV multiple times, shuffling the data before each iteration, providing more robustness in evaluation.
6. Hyperparameter Tuning and Model Selection:
    - K-Fold Cross-Validation is often used in hyperparameter tuning (using grid search or randomized search) to select the best model based on the average performance across folds**.**

6.2. Code:



```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
pd.set_option("display.max_columns", None)
from warnings import filterwarnings
filterwarnings("ignore",category=DeprecationWarning)
filterwarnings("ignore", category=FutureWarning)
filterwarnings("ignore", category=UserWarning)
import warnings
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
# plotly
import plotly.graph_objs as go
from plotly import tools
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.express as px
import math
```



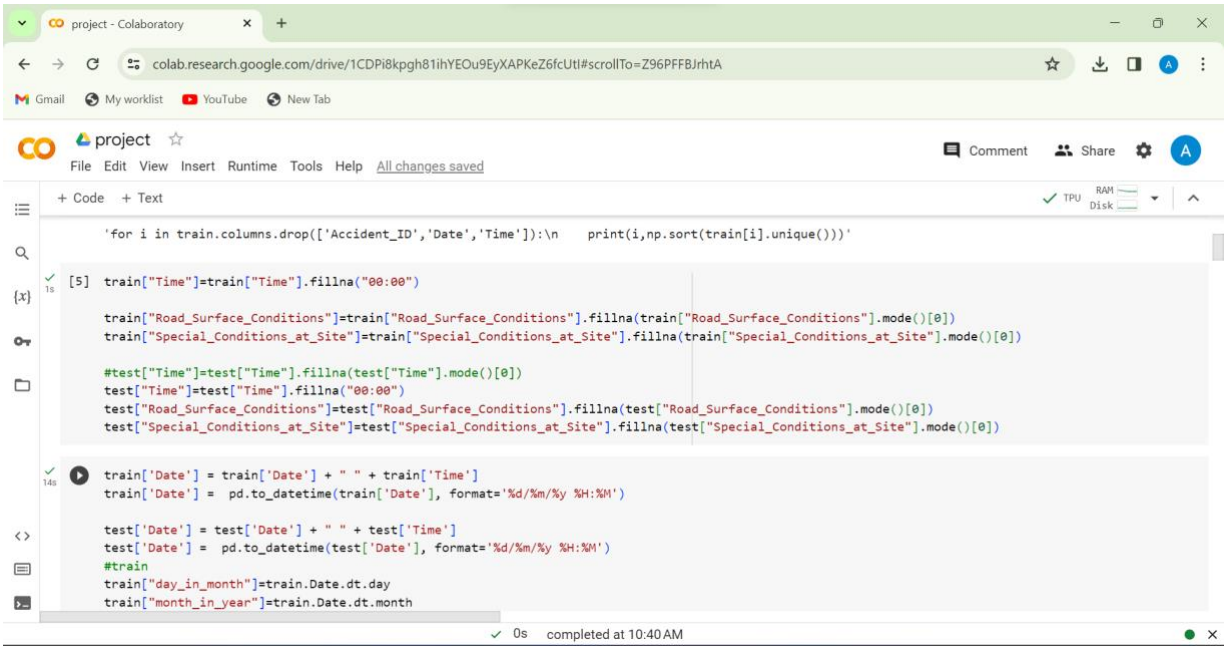
```
import math
import calendar
```

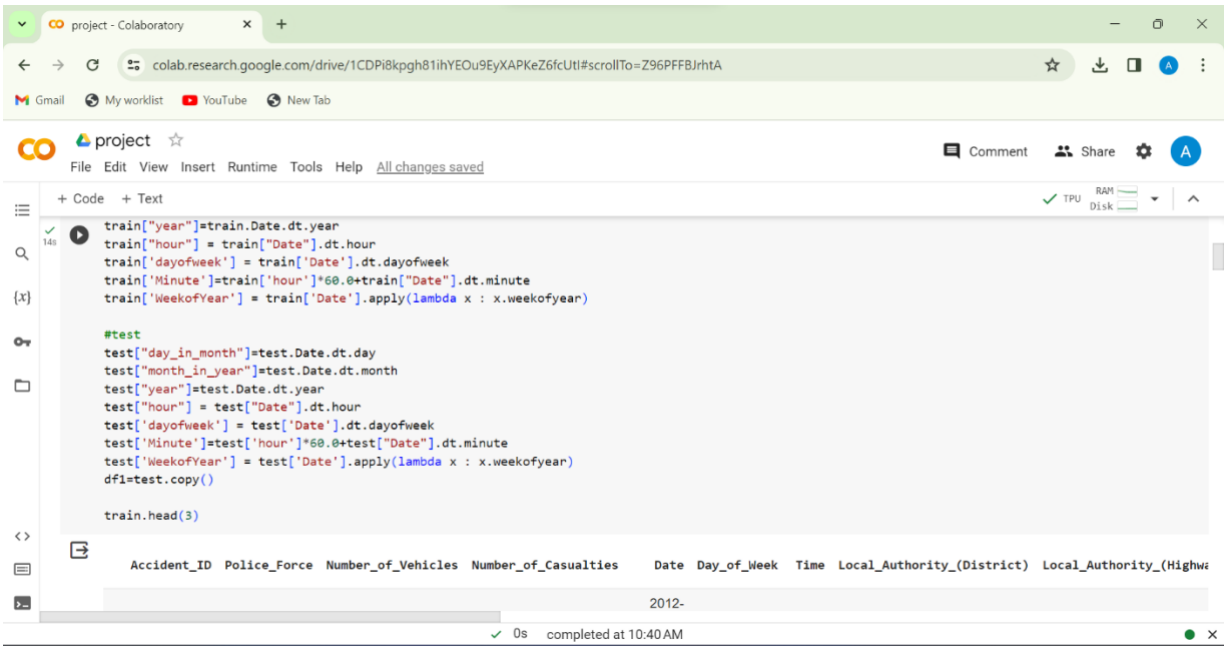
```
[2] train = pd.read_csv('/content/drive/MyDrive/project/train.csv')
    test  = pd.read_csv('/content/drive/MyDrive/project/test.csv')
    sample_submission  = pd.read_csv('/content/drive/MyDrive/project/sample_submission.csv')
```

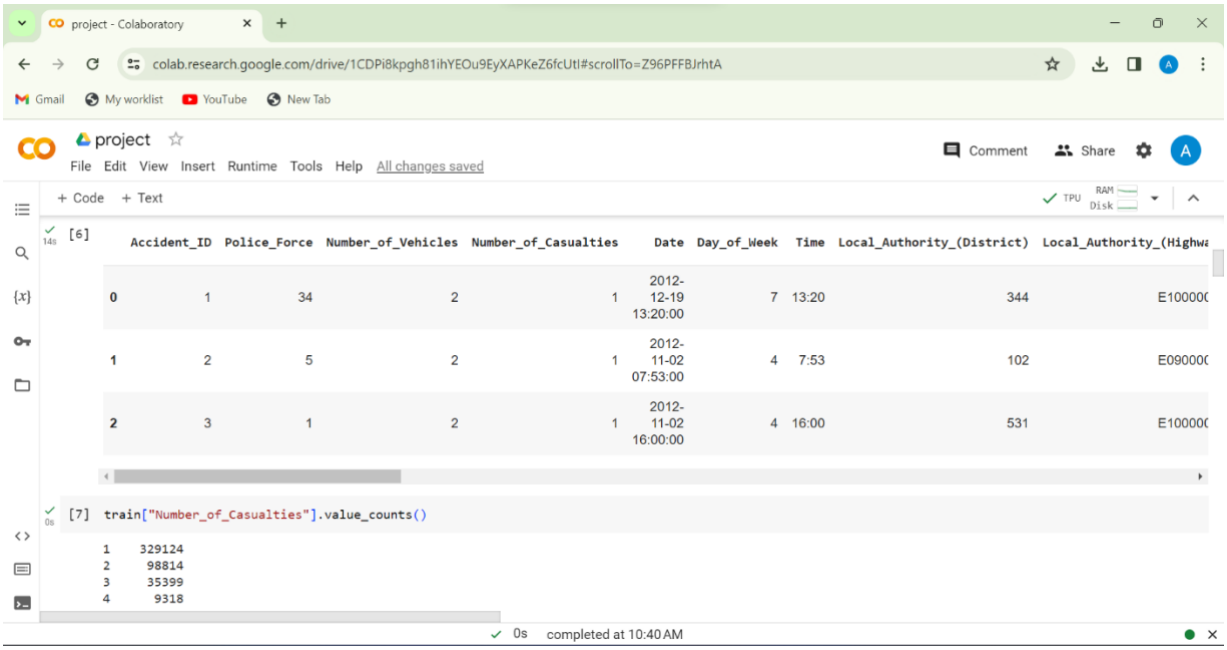
```
[3] print('There are {} rows and {} columns in train'.format(train.shape[0],train.shape[1]))
    print('There are {} rows and {} columns in test'.format(test.shape[0],test.shape[1]))
    print('There are {} rows and {} columns in sample_submission'.format(sample_submission.shape[0],sample_submission.shape[1]))
```

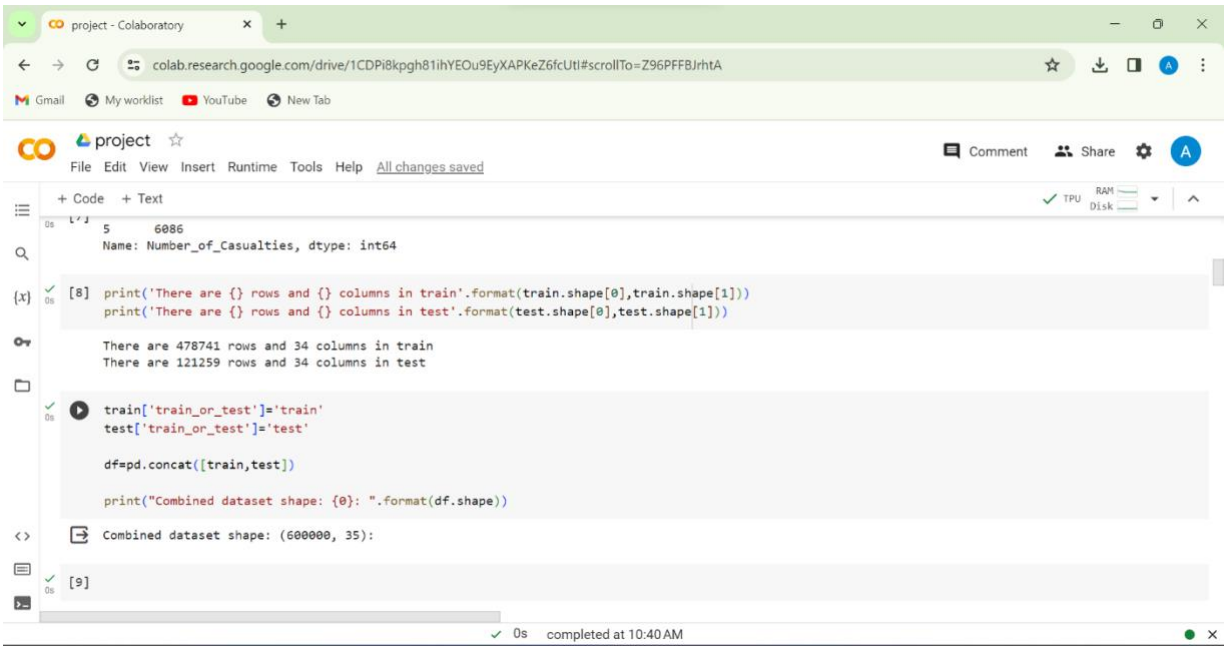
```
There are 478741 rows and 27 columns in train
There are 121259 rows and 27 columns in test
There are 49772 rows and 2 columns in sample_submission
```

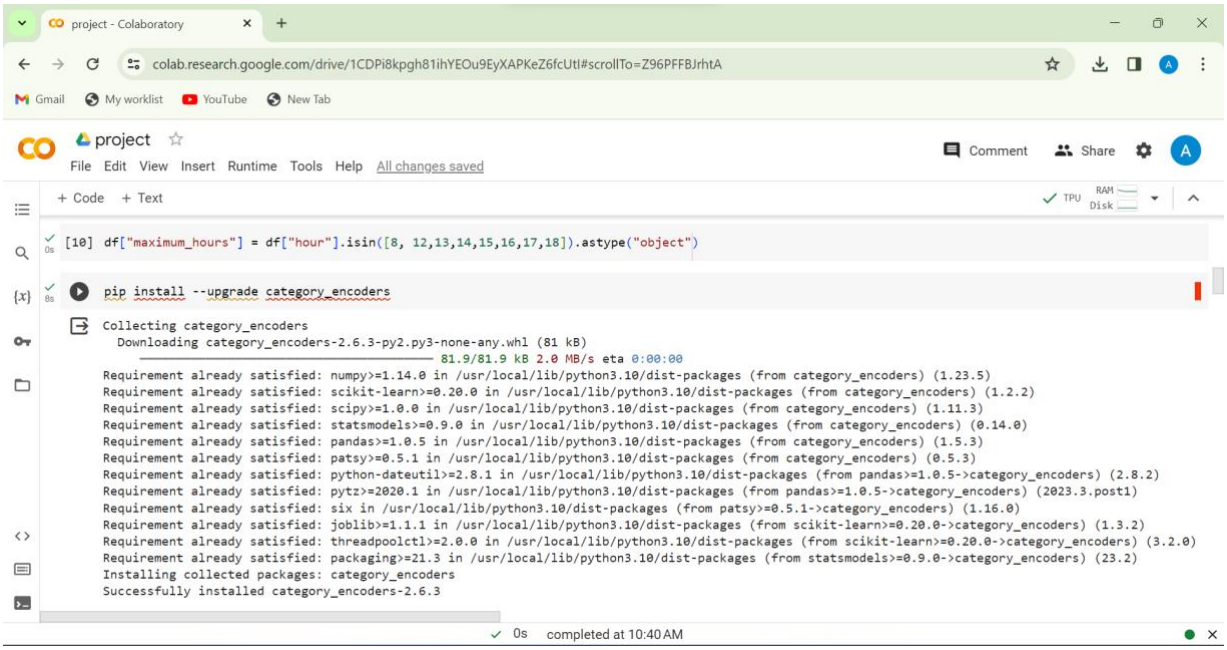
```
[4] #Check unique values of each column
    '''for i in train.columns.drop(['Accident_ID','Date','Time']):
        print(i,np.sort(train[i].unique()))'''
```

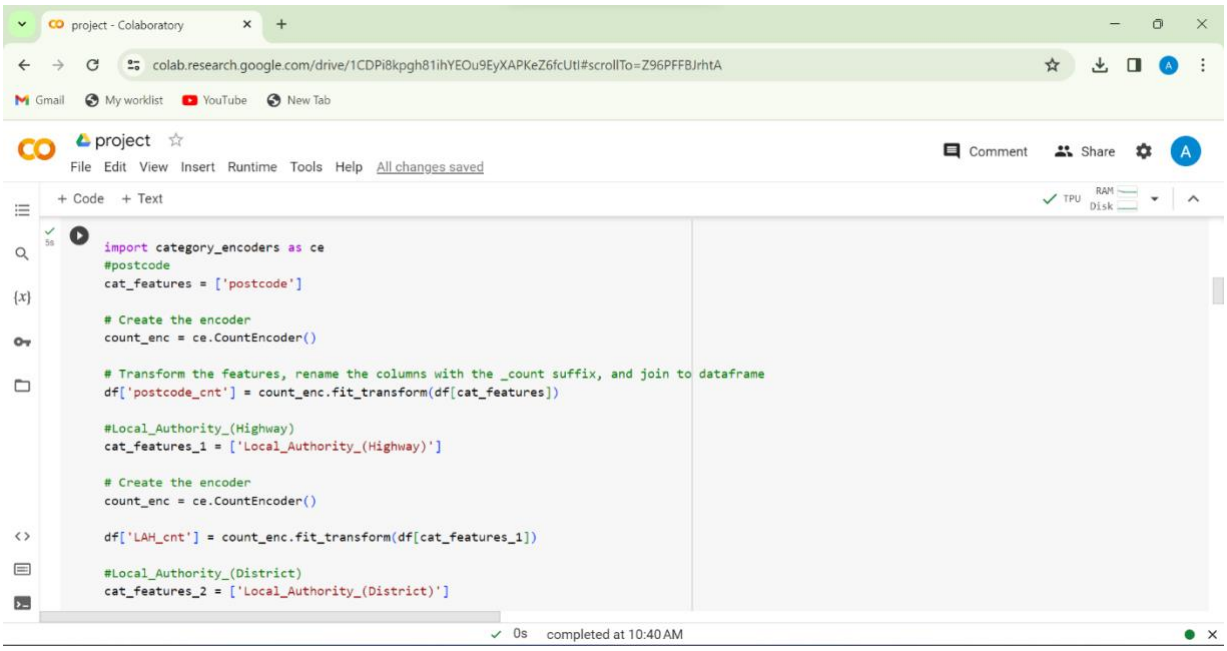

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
'for i in train.columns.drop(['Accident_ID','Date','Time']):\n print(i,np.sort(train[i].unique()))'
[5] train["Time"]=train["Time"].fillna("00:00")
train["Road_Surface_Conditions"]=train["Road_Surface_Conditions"].fillna(train["Road_Surface_Conditions"].mode()[0])
train["Special_Conditions_at_Site"]=train["Special_Conditions_at_Site"].fillna(train["Special_Conditions_at_Site"].mode()[0])
#test["Time"]=test["Time"].fillna(test["Time"].mode()[0])
test["Time"]=test["Time"].fillna("00:00")
test["Road_Surface_Conditions"]=test["Road_Surface_Conditions"].fillna(test["Road_Surface_Conditions"].mode()[0])
test["Special_Conditions_at_Site"]=test["Special_Conditions_at_Site"].fillna(test["Special_Conditions_at_Site"].mode()[0])
train['Date'] = train['Date'] + " " + train['Time']
train['Date'] = pd.to_datetime(train['Date'], format='%d/%m/%y %H:%M')
test['Date'] = test['Date'] + " " + test['Time']
test['Date'] = pd.to_datetime(test['Date'], format='%d/%m/%y %H:%M')
#train
train["day_in_month"]=train.Date.dt.day
train["month_in_year"]=train.Date.dt.month
0s completed at 10:40 AM

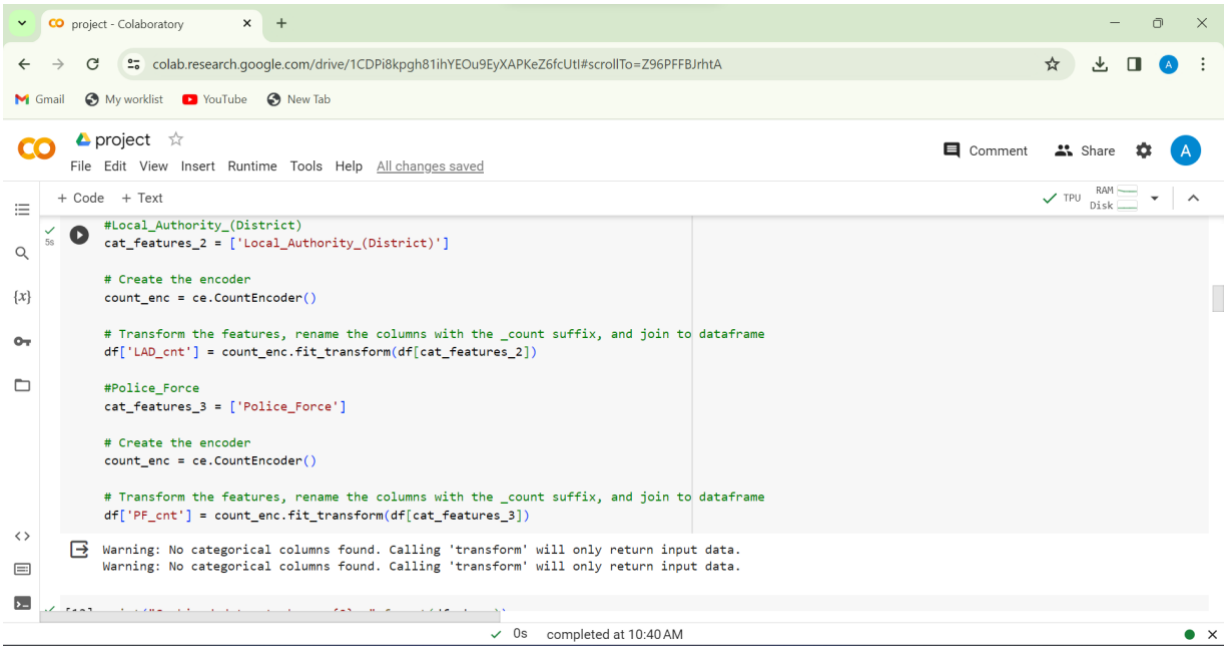

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
train["year"]=train.Date.dt.year
train["hour"] = train["Date"].dt.hour
train['dayofweek'] = train['Date'].dt.dayofweek
train['Minute']=train['hour']*60.0+train["Date"].dt.minute
train['WeekofYear'] = train['Date'].apply(lambda x : x.weekofyear)
#test
test["day_in_month"]=test.Date.dt.day
test["month_in_year"]=test.Date.dt.month
test["year"]=test.Date.dt.year
test["hour"] = test["Date"].dt.hour
test['dayofweek'] = test['Date'].dt.dayofweek
test['Minute']=test['hour']*60.0+test["Date"].dt.minute
test['WeekofYear'] = test['Date'].apply(lambda x : x.weekofyear)
df1=test.copy()
train.head(3)
Accident_ID Police_Force Number_of_Vehicles Number_of_Casualties Date Day_of_Week Time Local_Authority_(District) Local_Authority_(Highwa
2012-
0s completed at 10:40 AM

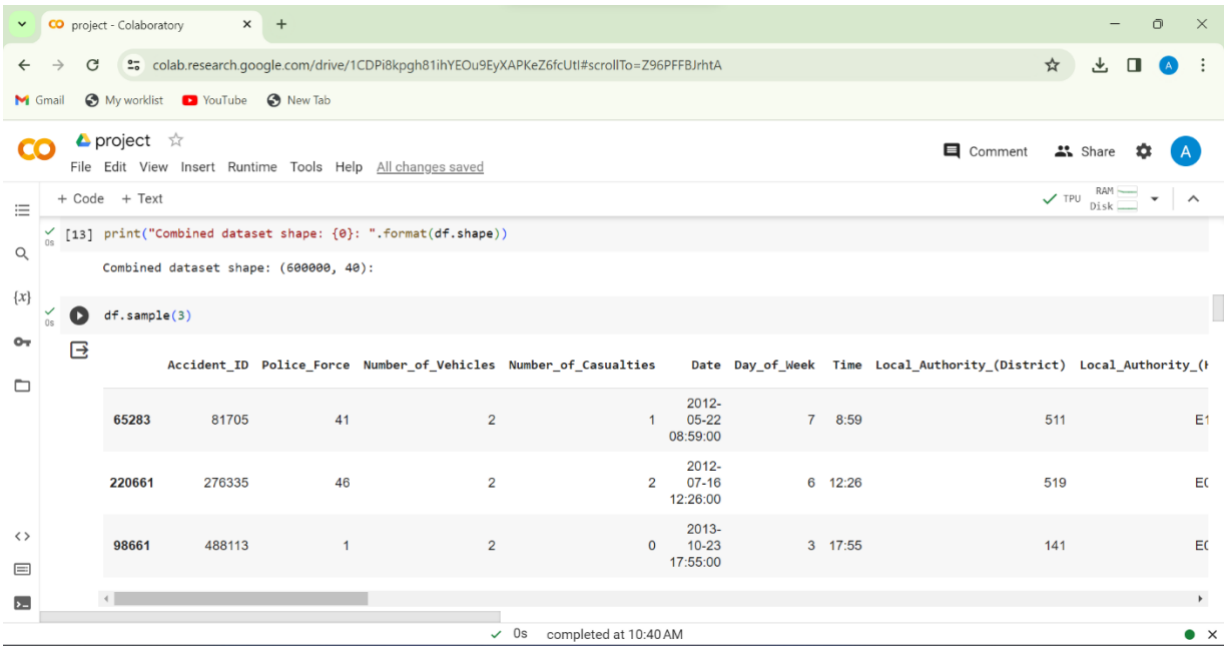

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
[6]
Accident_ID Police_Force Number_of_Vehicles Number_of_Casualties Date Day_of_Week Time Local_Authority_(District) Local_Authority_(Highwa
0 1 34 2 1 2012-12-19 13:20:00 7 13:20 344 E100000
1 2 5 2 1 2012-11-02 07:53:00 4 7:53 102 E090000
2 3 1 2 1 2012-11-02 16:00:00 4 16:00 531 E100000
[7]
train["Number_of_Casualties"].value_counts()
1 329124
2 98814
3 35399
4 9318
0s completed at 10:40 AM

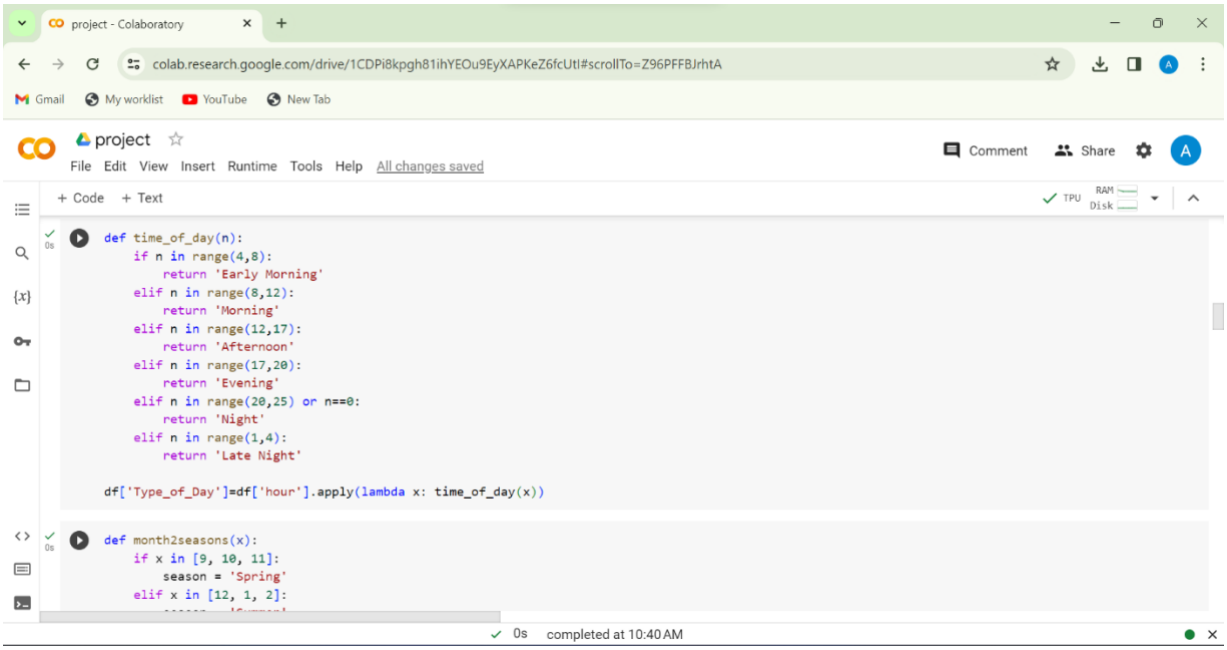

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
5 6086
Name: Number_of_Casualties, dtype: int64
[8]
print('There are {} rows and {} columns in train'.format(train.shape[0],train.shape[1]))
print('There are {} rows and {} columns in test'.format(test.shape[0],test.shape[1]))
There are 478741 rows and 34 columns in train
There are 121259 rows and 34 columns in test
train['train_or_test']='train'
test['train_or_test']='test'
df=pd.concat([train,test])
print("Combined dataset shape: {0}: ".format(df.shape))
Combined dataset shape: (600000, 35):
[9]
0s completed at 10:40 AM

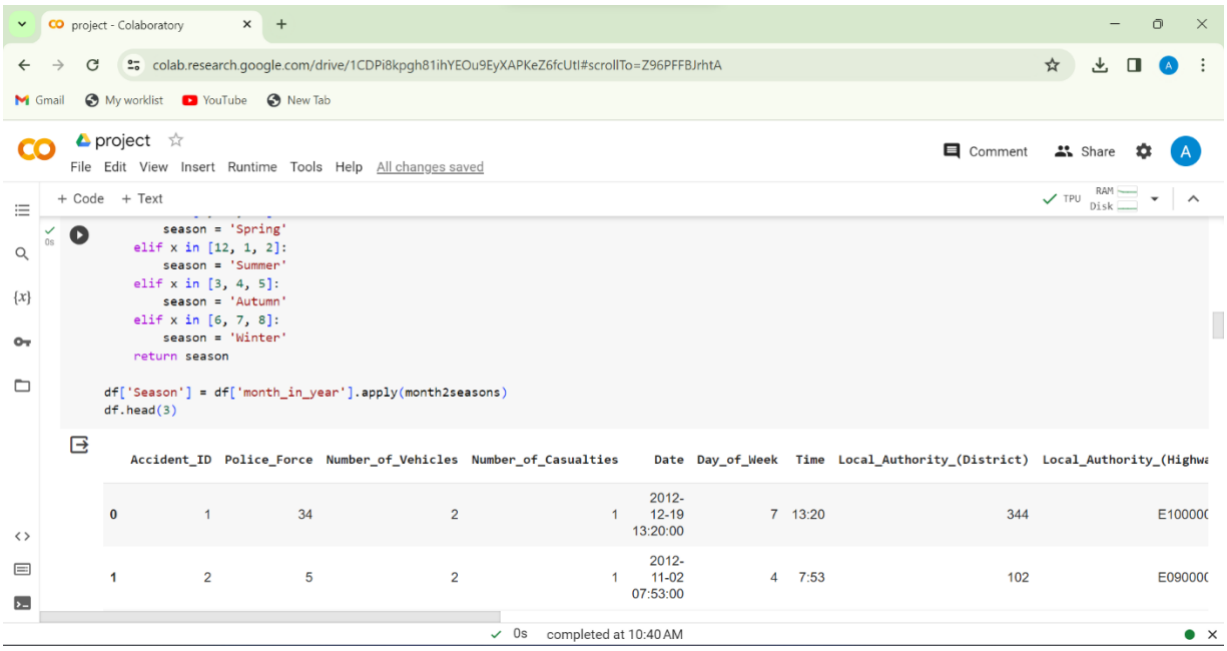

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
[10] df["maximum_hours"] = df["hour"].isin([8, 12,13,14,15,16,17,18]).astype("object")
pip install --upgrade category_encoders
Collecting category_encoders
Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
81.9/81.9 kB 2.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.23.5)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.2.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.11.3)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.14.0)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.5.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2023.3.post1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encoders) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encoders) (3.2.0)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->category_encoders) (23.2)
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.3
0s completed at 10:40 AM

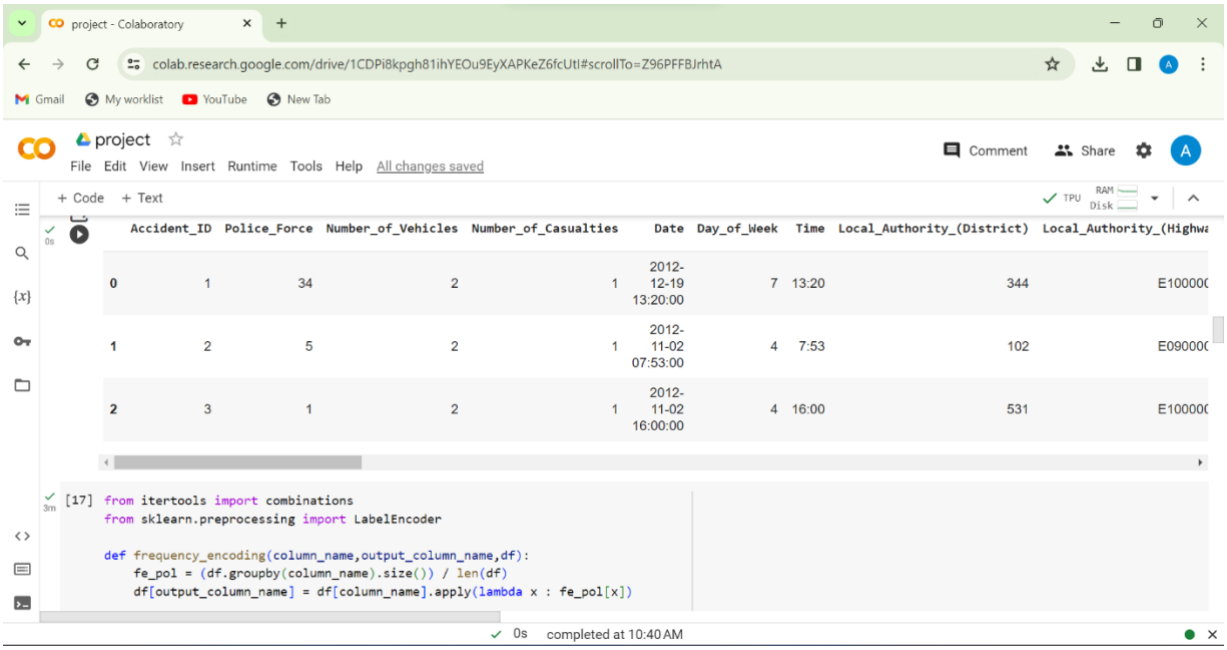

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
import category_encoders as ce
#postcode
cat_features = ['postcode']
# Create the encoder
count_enc = ce.CountEncoder()
# Transform the features, rename the columns with the _count suffix, and join to dataframe
df['postcode_cnt'] = count_enc.fit_transform(df[cat_features])
#Local_Authority_(Highway)
cat_features_1 = ['Local_Authority_(Highway)']
# Create the encoder
count_enc = ce.CountEncoder()
df['LAH_cnt'] = count_enc.fit_transform(df[cat_features_1])
#Local_Authority_(District)
cat_features_2 = ['Local_Authority_(District)']
0s completed at 10:40 AM

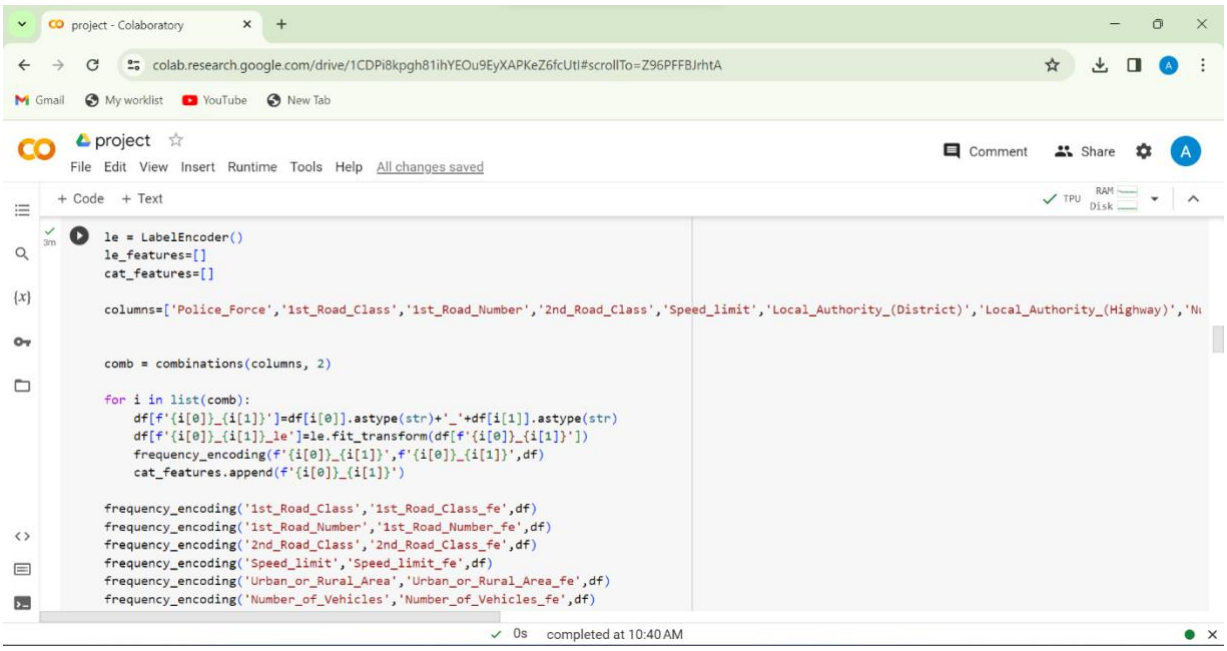

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
#Local_Authority_(District)
cat_features_2 = ['Local_Authority_(District)']
# Create the encoder
count_enc = ce.CountEncoder()
# Transform the features, rename the columns with the _count suffix, and join to dataframe
df['LAD_cnt'] = count_enc.fit_transform(df[cat_features_2])
#Police_Force
cat_features_3 = ['Police_Force']
# Create the encoder
count_enc = ce.CountEncoder()
# Transform the features, rename the columns with the _count suffix, and join to dataframe
df['PF_cnt'] = count_enc.fit_transform(df[cat_features_3])
Warning: No categorical columns found. Calling 'transform' will only return input data.
Warning: No categorical columns found. Calling 'transform' will only return input data.
0s completed at 10:40 AM

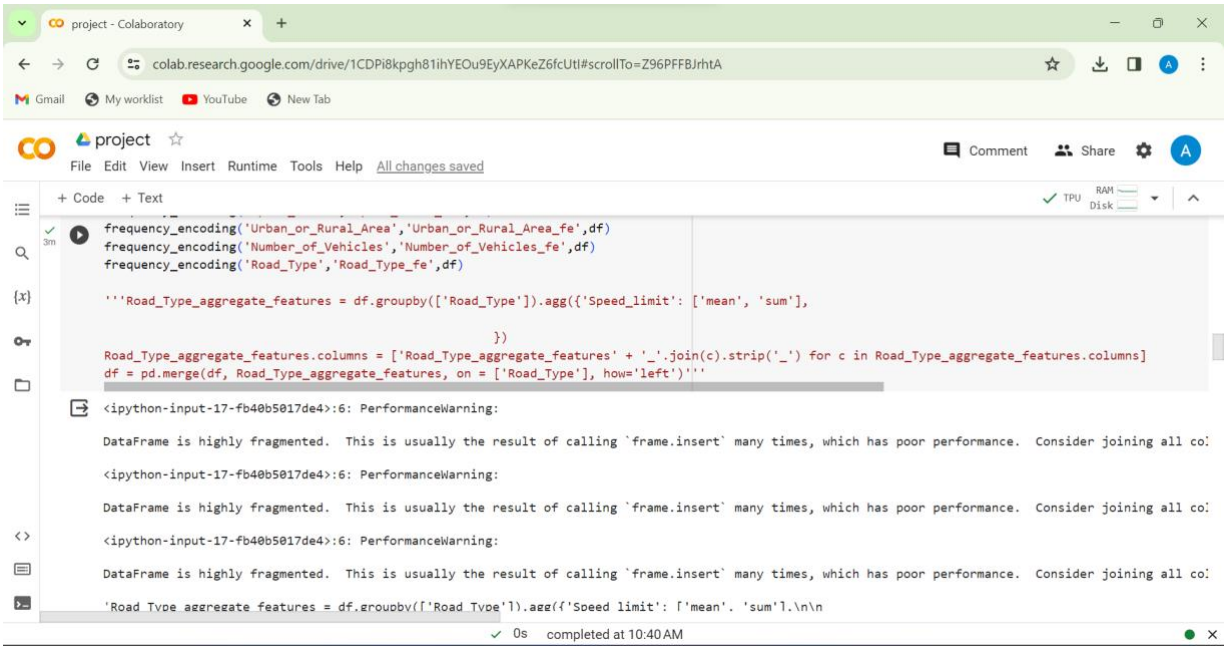

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
[13] print("Combined dataset shape: {0}: ".format(df.shape))
Combined dataset shape: (600000, 40):
df.sample(3)
Accident_ID Police_Force Number_of_Vehicles Number_of_Casualties Date Day_of_Week Time Local_Authority_(District) Local_Authority_(
65283 81705 41 2 1 2012-05-22 08:59:00 7 8:59 511 E1
220661 276335 46 2 2 2012-07-16 12:26:00 6 12:26 519 E0
98661 488113 1 2 0 2013-10-23 17:55:00 3 17:55 141 E0
0s completed at 10:40 AM

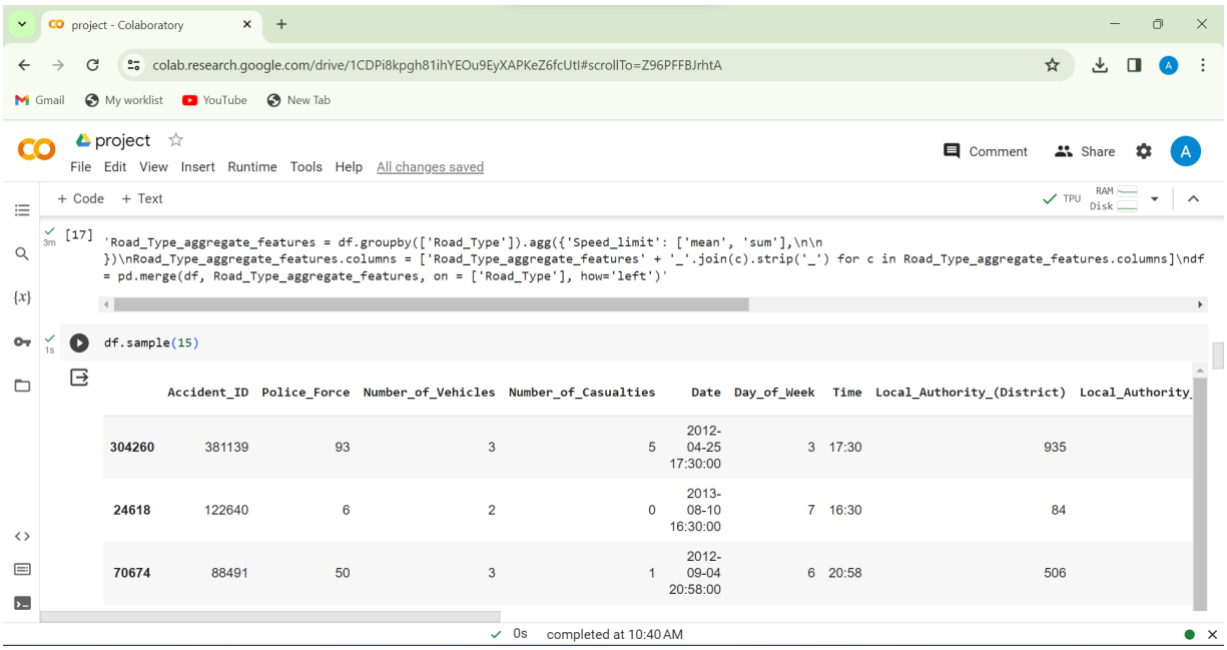

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
def time_of_day(n):
if n in range(4,8):
return 'Early Morning'
elif n in range(8,12):
return 'Morning'
elif n in range(12,17):
return 'Afternoon'
elif n in range(17,20):
return 'Evening'
elif n in range(20,25) or n==0:
return 'Night'
elif n in range(1,4):
return 'Late Night'
df['Type_of_Day']=df['hour'].apply(lambda x: time_of_day(x))
def month2seasons(x):
if x in [9, 10, 11]:
season = 'Spring'
elif x in [12, 1, 2]:
0s completed at 10:40 AM

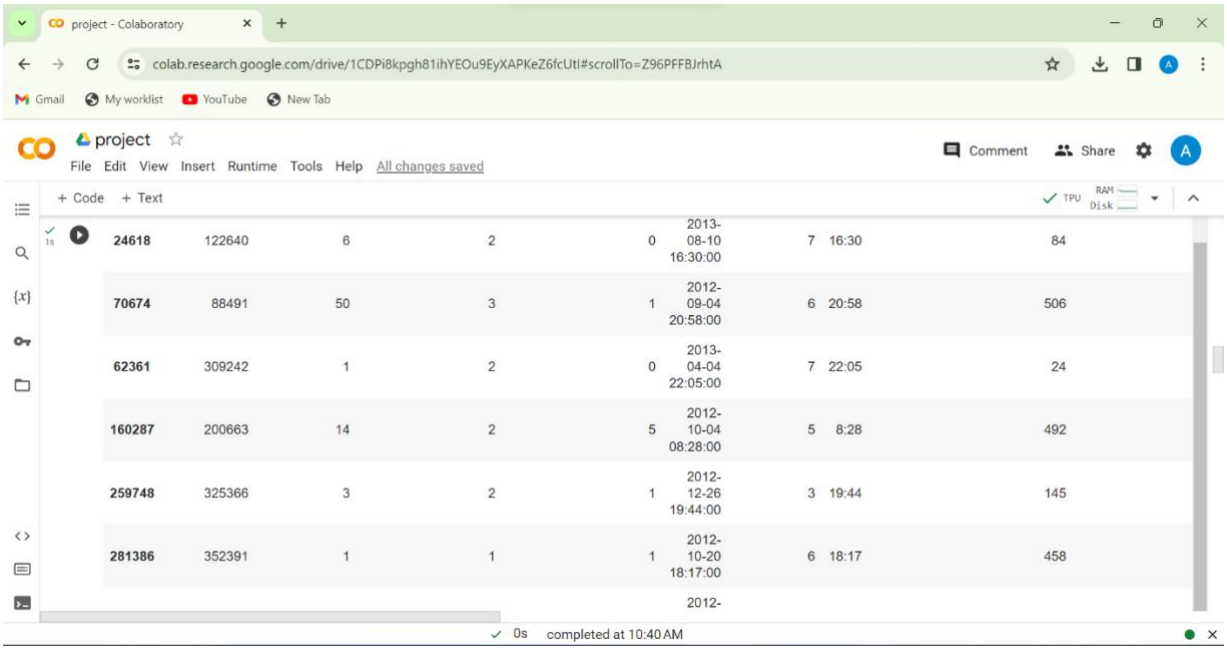

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
season = 'Spring'
elif x in [12, 1, 2]:
season = 'Summer'
elif x in [3, 4, 5]:
season = 'Autumn'
elif x in [6, 7, 8]:
season = 'Winter'
return season
df['Season'] = df['month_in_year'].apply(month2seasons)
df.head(3)
Accident_ID Police_Force Number_of_Vehicles Number_of_Casualties Date Day_of_Week Time Local_Authority_(District) Local_Authority_(Highwa
0 1 34 2 1 2012-12-19 13:20:00 7 13:20 344 E100000
1 2 5 2 1 2012-11-02 07:53:00 4 7:53 102 E090000
0s completed at 10:40 AM

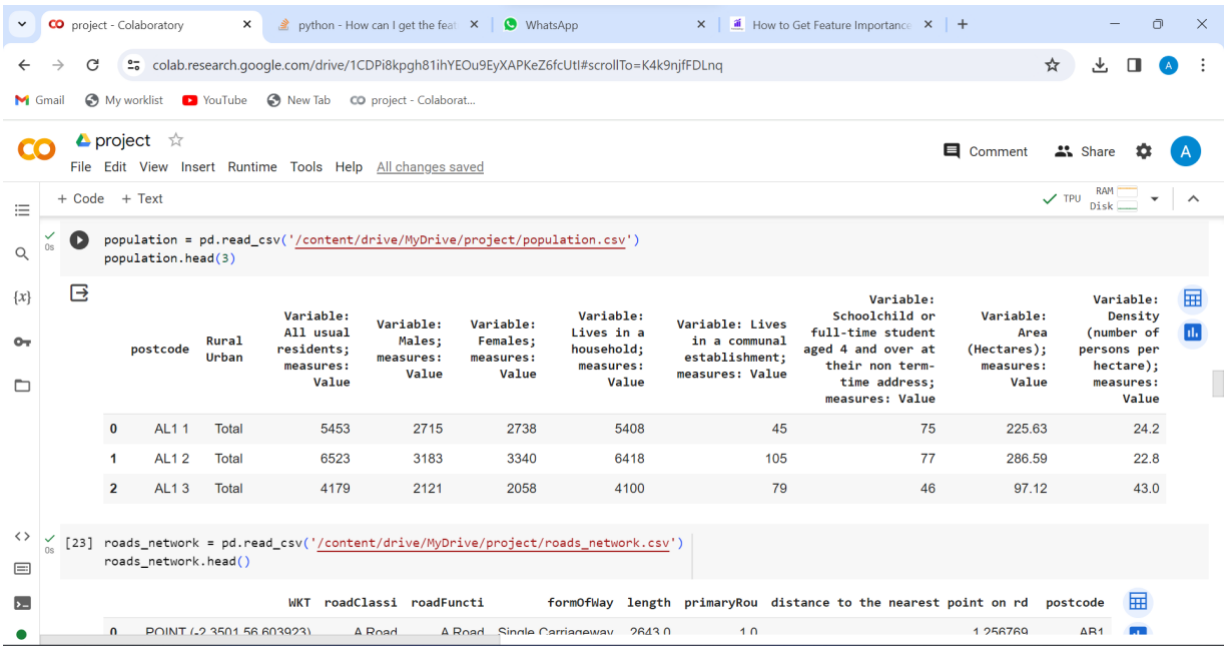

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
Accident_ID Police_Force Number_of_Vehicles Number_of_Casualties Date Day_of_Week Time Local_Authority_(District) Local_Authority_(Highwa
0 1 34 2 1 2012-12-19 13:20:00 7 13:20 344 E10000
1 2 5 2 1 2012-11-02 07:53:00 4 7:53 102 E09000
2 3 1 2 1 2012-11-02 16:00:00 4 16:00 531 E10000
[17] from itertools import combinations
from sklearn.preprocessing import LabelEncoder
def frequency_encoding(column_name,output_column_name,df):
fe_pol = (df.groupby(column_name).size()) / len(df)
df[output_column_name] = df[column_name].apply(lambda x : fe_pol[x])
0s completed at 10:40 AM

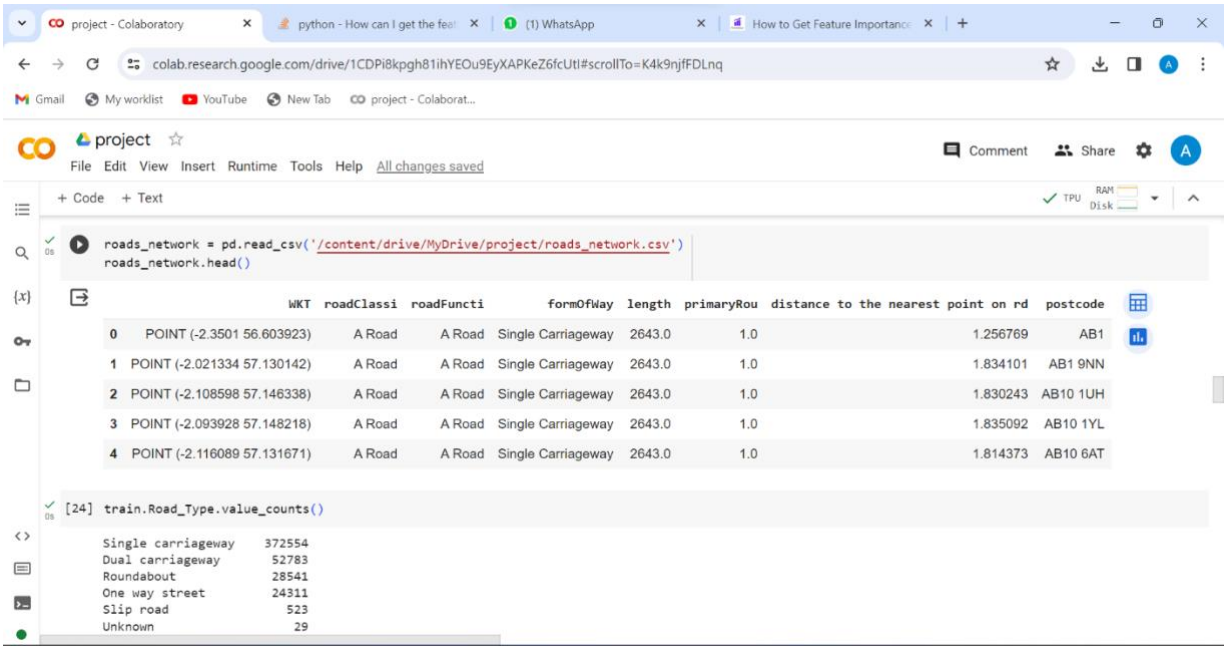

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
le = LabelEncoder()
le_features=[]
cat_features=[]
columns=['Police_Force','1st_Road_Class','1st_Road_Number','2nd_Road_Class','Speed_limit','Local_Authority_(District)','Local_Authority_(Highway)','Nu
comb = combinations(columns, 2)
for i in list(comb):
df[f'{i[0]}_{i[1]}']=df[i[0]].astype(str)+'_'+df[i[1]].astype(str)
df[f'{i[0]}_{i[1]}_le']=le.fit_transform(df[f'{i[0]}_{i[1]}'])
frequency_encoding(f'{i[0]}_{i[1]}',f'{i[0]}_{i[1]}',df)
cat_features.append(f'{i[0]}_{i[1]}')
frequency_encoding('1st_Road_Class','1st_Road_Class_fe',df)
frequency_encoding('1st_Road_Number','1st_Road_Number_fe',df)
frequency_encoding('2nd_Road_Class','2nd_Road_Class_fe',df)
frequency_encoding('Speed_limit','Speed_limit_fe',df)
frequency_encoding('Urban_or_Rural_Area','Urban_or_Rural_Area_fe',df)
frequency_encoding('Number_of_Vehicles','Number_of_Vehicles_fe',df)
0s completed at 10:40 AM


project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
frequency_encoding('Urban_or_Rural_Area','Urban_or_Rural_Area_fe',df)
frequency_encoding('Number_of_Vehicles','Number_of_Vehicles_fe',df)
frequency_encoding('Road_Type','Road_Type_fe',df)
'''Road_Type_aggregate_features = df.groupby(['Road_Type']).agg({'Speed_limit': ['mean', 'sum'],
})
Road_Type_aggregate_features.columns = ['Road_Type_aggregate_features' + '_'.join(c).strip('_') for c in Road_Type_aggregate_features.columns]
df = pd.merge(df, Road_Type_aggregate_features, on = ['Road_Type'], how='left')'''
<ipython-input-17-fb40b5017de4>:6: PerformanceWarning:
DataFrame is highly fragmented. This is usually the result of calling `frame.insert` many times, which has poor performance. Consider joining all co
<ipython-input-17-fb40b5017de4>:6: PerformanceWarning:
DataFrame is highly fragmented. This is usually the result of calling `frame.insert` many times, which has poor performance. Consider joining all co
<ipython-input-17-fb40b5017de4>:6: PerformanceWarning:
DataFrame is highly fragmented. This is usually the result of calling `frame.insert` many times, which has poor performance. Consider joining all co
'Road_Type_aggregate_features = df.groupby(['Road_Type']).agg({'Speed_limit': ['mean', 'sum'],\n\n
0s completed at 10:40 AM

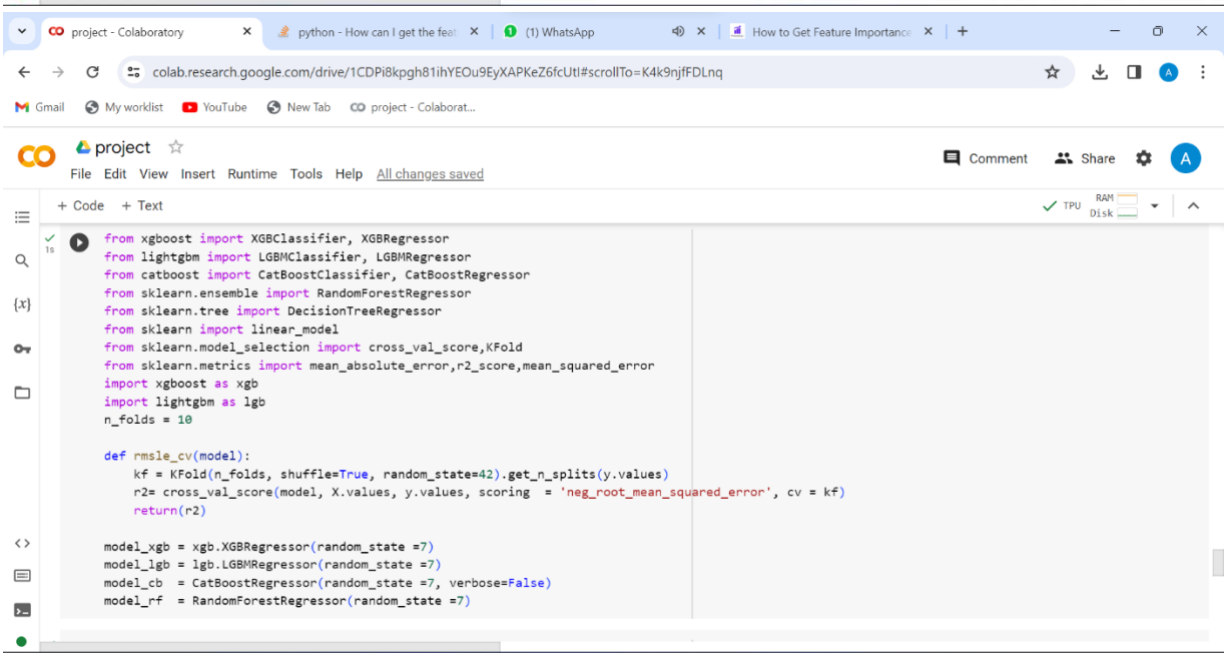

project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
[17] 'Road_Type_aggregate_features = df.groupby(['Road_Type']).agg({'Speed_limit': ['mean', 'sum'],\n\n
})\nRoad_Type_aggregate_features.columns = ['Road_Type_aggregate_features' + '_'.join(c).strip('_') for c in Road_Type_aggregate_features.columns]\ndf
= pd.merge(df, Road_Type_aggregate_features, on = ['Road_Type'], how='left')'
df.sample(15)
Accident_ID Police_Force Number_of_Vehicles Number_of_Casualties Date Day_of_Week Time Local_Authority_(District) Local_Authority
304260 381139 93 3 5 2012-04-25 17:30:00 3 17:30 935
24618 122640 6 2 0 2013-08-10 16:30:00 7 16:30 84
70674 88491 50 3 1 2012-09-04 20:58:00 6 20:58 506
0s completed at 10:40 AM


project - Colaboratory
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=Z96PFFBJrhtA
Gmail
My worklist
YouTube
New Tab
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
24618 122640 6 2 0 2013-08-10 16:30:00 7 16:30 84
70674 88491 50 3 1 2012-09-04 20:58:00 6 20:58 506
62361 309242 1 2 0 2013-04-04 22:05:00 7 22:05 24
160287 200663 14 2 5 2012-10-04 08:28:00 5 8:28 492
259748 325366 3 2 1 2012-12-26 19:44:00 3 19:44 145
281386 352391 1 1 1 2012-10-20 18:17:00 6 18:17 458
2012-
0s completed at 10:40 AM


project - Colaboratory
python - How can I get the feat
WhatsApp
How to Get Feature Importance
colab.research.google.com/drive/1CDPi8kpgh81ihYEOu9EyXAPKeZ6fcUtI#scrollTo=K4k9njfFDLnq
Gmail
My worklist
YouTube
New Tab
project - Colaborat...
project
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
TPU
RAM
Disk
population = pd.read_csv('/content/drive/MyDrive/project/population.csv')
population.head(3)
postcode Rural Urban Variable: All usual residents; measures: Value Variable: Males; measures: Value Variable: Females; measures: Value Variable: Lives in a household; measures: Value Variable: Lives in a communal establishment; measures: Value Variable: Schoolchild or full-time student aged 4 and over at their non term-time address; measures: Value Variable: Area (Hectares); measures: Value Variable: Density (number of persons per hectare); measures: Value
0 AL1 1 Total 5453 2715 2738 5408 45 75 225.63 24.2
1 AL1 2 Total 6523 3183 3340 6418 105 77 286.59 22.8
2 AL1 3 Total 4179 2121 2058 4100 79 46 97.12 43.0
[23] roads_network = pd.read_csv('/content/drive/MyDrive/project/roads_network.csv')
roads_network.head()
WKT roadClassi roadFuncti formOfWay length primaryRou distance to the nearest point on rd postcode
0 POINT (-2.3501 56.603923) A Road A Road Single Carriageway 2643.0 1.0 1.256769 AB1



	WKT	roadClassi	roadFuncti	formOfWay	length	primaryRou	distance to the nearest point on rd	postcode
0	POINT (-2.3501 56.603923)	A Road	A Road	Single Carriageway	2643.0	1.0	1.256769	AB1
1	POINT (-2.021334 57.130142)	A Road	A Road	Single Carriageway	2643.0	1.0	1.834101	AB1 9NN
2	POINT (-2.108598 57.146338)	A Road	A Road	Single Carriageway	2643.0	1.0	1.830243	AB10 1UH
3	POINT (-2.093928 57.148218)	A Road	A Road	Single Carriageway	2643.0	1.0	1.835092	AB10 1YL
4	POINT (-2.116089 57.131671)	A Road	A Road	Single Carriageway	2643.0	1.0	1.814373	AB10 6AT



7.SOFTWARE TESTING

7.1 INTRODUCTION:

The suppliers and products used to implement components are displayed at the implementation level. The diagrams of the logical levels of the architecture form each visualization of this page. An architectural diagram often includes a mixture of logical and physical aspects. An architectural diagram is a graphic representation of each component that forms a system, whether fully or partially. Above all, it helps to understand the layout of a system or application by engineers, designers, stakeholders and everyone else.

Programme testing aims to find bugs in the existing programme as well as approaches to improve the software's efficiency, precision, and usefulness. It basically aims to evaluate the specification, functionality, and performance of a piece of software or an application.

There are two stages to software testing:

1. Verification: This is the group of procedures that make sure the program carries out a certain function appropriately.
2. Validation: This term refers to a variety of procedures that guarantee the software is traceable to client requirements.

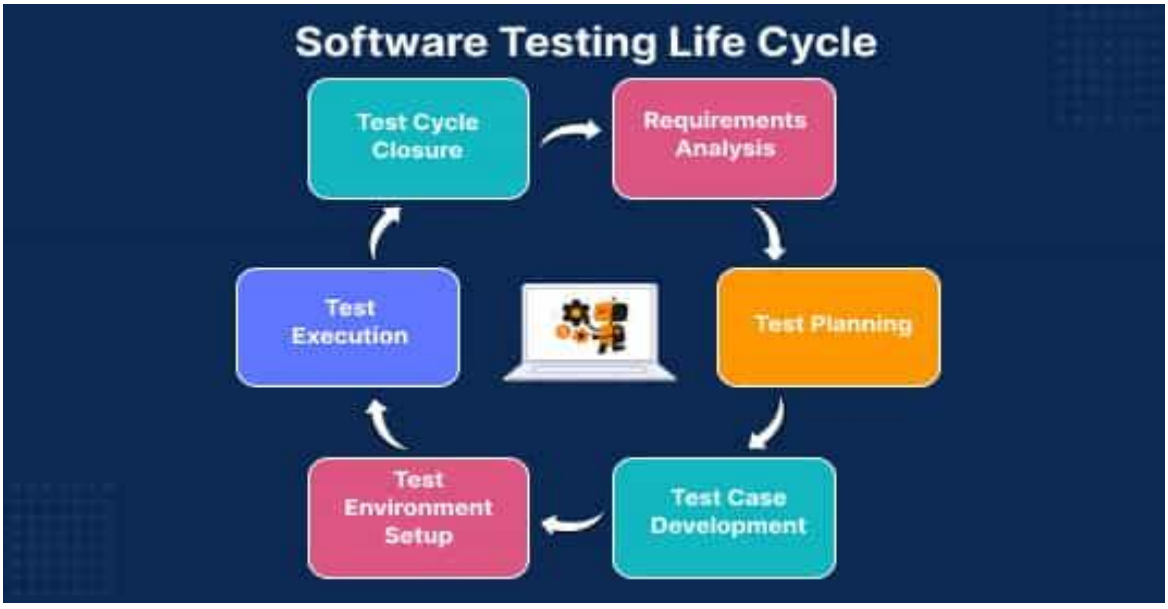


Fig 15: Software testing lifecycle

7.1.1 UNIT TESTING:

A unit is a single test component that can be tested as part of the application software development process. To verify that the isolated code is correct, unit tests are conducted. A single function or code block is called a unit component. Functional testing starts with unit testing.

In the hierarchy of test levels, unit tests are first, followed by integration tests and higher test levels. It uses testing modules to reduce waiting times for Stubs, drivers, dummy objects and unit testing frameworks that are useful for unit testing.

Unit tests, integration tests, system tests and acceptance tests are the four levels of software usually tested. However, software testers sometimes abandon unit tests due to time constraints, leading to more errors during integration tests, system tests, acceptance tests, or even beta tests after completing a software application.

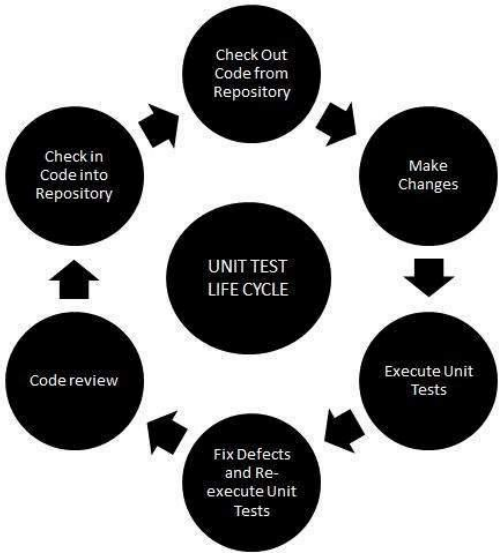


Fig 16: Unit testing

7.1.2 INTEGRATION TESTING:

After unit testing, the software testing process moves to integration testing. Units or individual software components are checked collectively during this test. Components are used in unit tests for testing purposes, while integration tests combine and test these components. Software is created using a variety of software modules created by different programmers and encoders. Integrity tests are performed to ensure that all modules communicate properly.

After each application module has been tested functionally, we will then go on to the integration testing. In order to ensure that the appropriate sequence is followed and that there is no missing integration case, we always perform integration tests by selecting the modules at the same time. First, determine the approach of the test case, which helps you create executable test cases based on test data. Examine the structure and design of the application to determine the most important modules to be tested first and to discover all potential situations. Create test cases to thoroughly test each interface. Select the input data before the test case is executed. Tests are largely dependent on input data. We should notify the developers of any defects that we identify so that they can be corrected.

After each application module has been functionally tested, we will only continue the integration tests. In order to ensure that the appropriate sequence is followed and that no integration cases are missed, we always perform integration tests by selecting modules one at a time. Determine first the approach of the test case, which will help you create executable test cases based on test data.

Examine the structure and design of the application to determine the most important modules to test first and discover all potential situations. Create test cases to thoroughly test each interface.

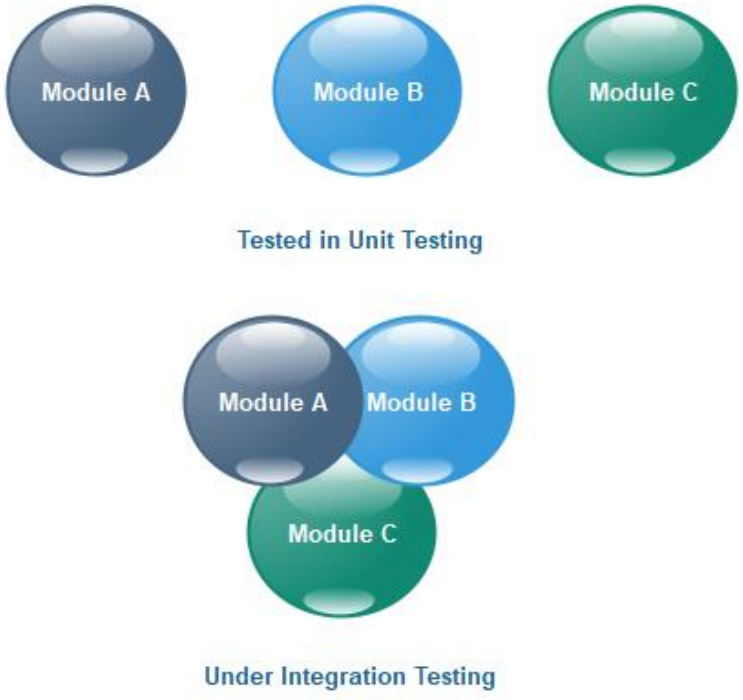


Fig 17: Integration testing

7.1.3 ACCEPTANCE TESTING:

Acceptance tests are a type of formal tests based on function processing and user requirements. The user determines whether the program meets the requirements of the user. The system acceptance level is tested in a similar way to a black box test with the required number of users.

The fourth and final stage of software testing is this phase. Before adopting the final product, the customer conducts a user acceptability test (UAT). UAT is often performed by the client (domain expert) to ensure satisfaction and determine whether the application works in accordance with the business scenarios provided in real time.

Acceptance testing can appear unnecessary once the product has gone through Unit Testing, Integration Testing, and System Testing, but it is necessary for the reasons listed below.

- If requirements change while a project is being developed, it may not be adequately conveyed to the development team.
- Developers create functions based on their own interpretation of the requirement document, and they might not be aware of the client's true requirements.
- Acceptance testing is crucial to detect these tiny faults since there may be some that can only be found when the system is actually utilized by the end user.

Different people can undertake the acceptability testing in various scenarios. For instance, the blue-dart firm asks TCS to create an application, and TCS accepts the request and agrees to deliver the program in two releases.

7.1.4 BLACK BOX TESTING:

Black box testing that examines only the functioning of the software without looking at its code . A customer-declared requirement specification is the main source of black box testing. In this approach, the tester selects a function, provides an input value to test its operation, and determines whether the function gives the desired result. If the output is accurate, the function passes the test and fails. The test team reviews the results with the development team and then moves to the next test function. If there are serious problems after all the functions have been tested, the development team will contact to make the necessary corrections.

Black box testing is a type of testing technique where the tester creates test cases to validate the functioning of the program while possessing detailed knowledge of how the product functions.

Software programming skills are not required. Each test case is created taking into account the input and output of a specific function. The tester only knows the specific result of a particular input, and they do not know the process of creating the result. Black box testing uses a variety of different test methods, including decision table techniques, boundary value analysis techniques, state transition techniques, all pair testing methods, cause-effect graph methods, equivalence partitioning methods, error guessing methods, use case methods, and user history methods. These strategies have all been well discussed.

The creation of test cases takes into account the definition of requirements. These test cases are generally constructed based on software work descriptions, including requirements, design criteria, and other specifications. To achieve a correct result, test designers choose positive test scenarios that have a valid input value and negative test scenarios that have an inaccurate input value. They can also be used for non-functional testing, but most test cases are created for functional testing. The testing team creates test cases and the software development team is not involved in this process.

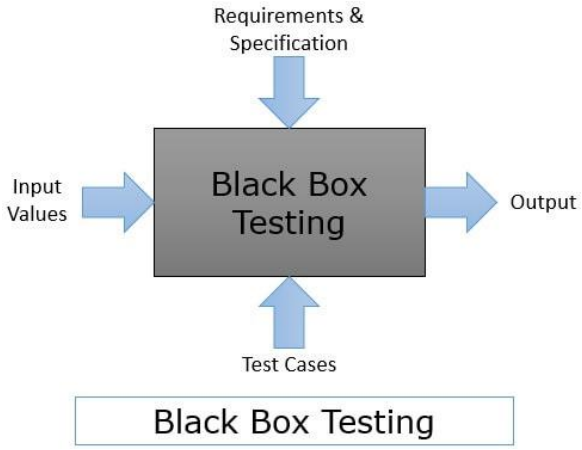


Fig 1: Blackbox testing

7.1.5 WHITE BOX TESTING:

Black box testing and white box testing are both part of the box testing methodology for software testing. In this article, we will talk about white box tests, also known as structural tests, clear box tests, open box tests and transparent box tests. It examines the core code and architecture of software, focusing on comparing predetermined inputs with expected and expected results. It is based on the internal operation of an application and focuses on internal structural testing. The creation of test cases for this kind of test requires programming knowledge. The main objectives

of white box testing are to focus on inputs and outputs through the program and to improve security.

Due to the internal perspective of the system, the word "white box" is used. The names "clear box", "white box" and "transparent box" refer to the transparency of the internal work of the software.

White Box tests are performed by developers. Each line of the program code is tested by the developer. The testing team performs black box tests after the developers have completed white box tests on the application or program, then they send them back to the developers after identifying any deficiencies. The developer delivers it to the test team after fixing the errors and conducting a series of white box tests.

The white box testing contains various tests, which are as follows:

- Path testing
- Loop testing
- Condition testing
- Test performance of the program
- Testing based on memory perspective

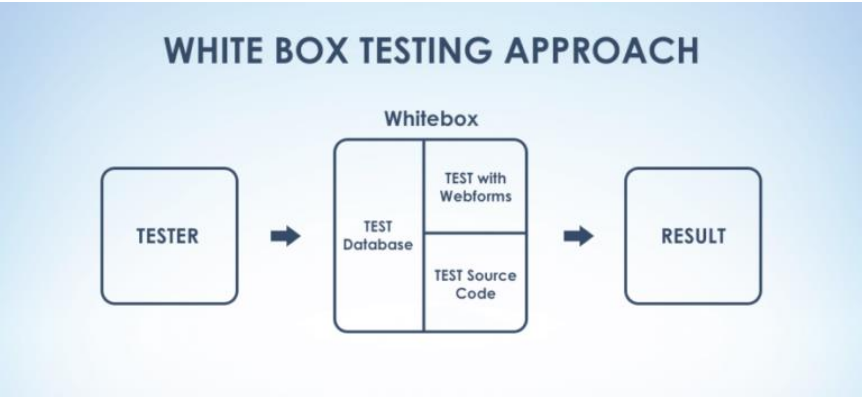
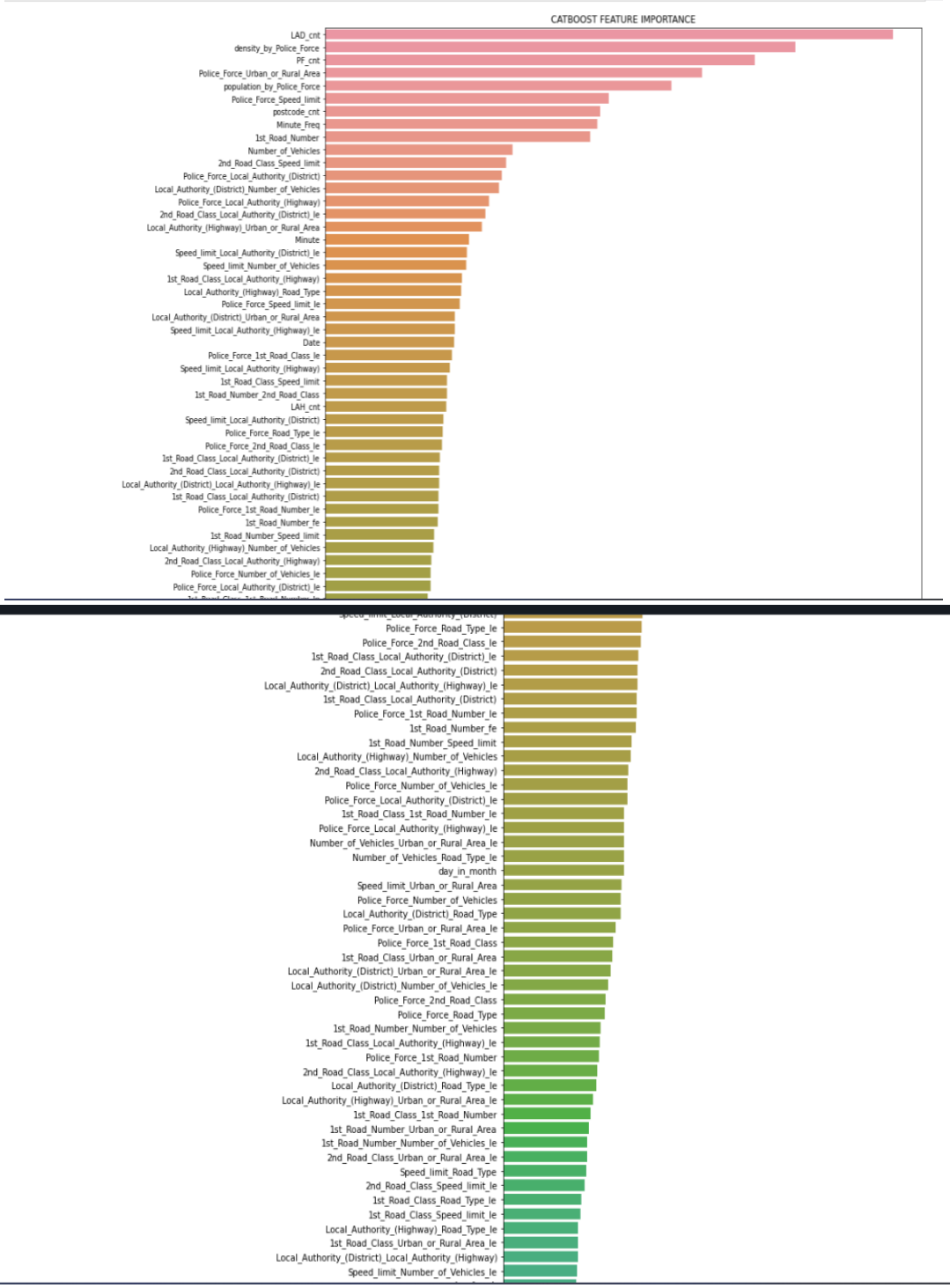


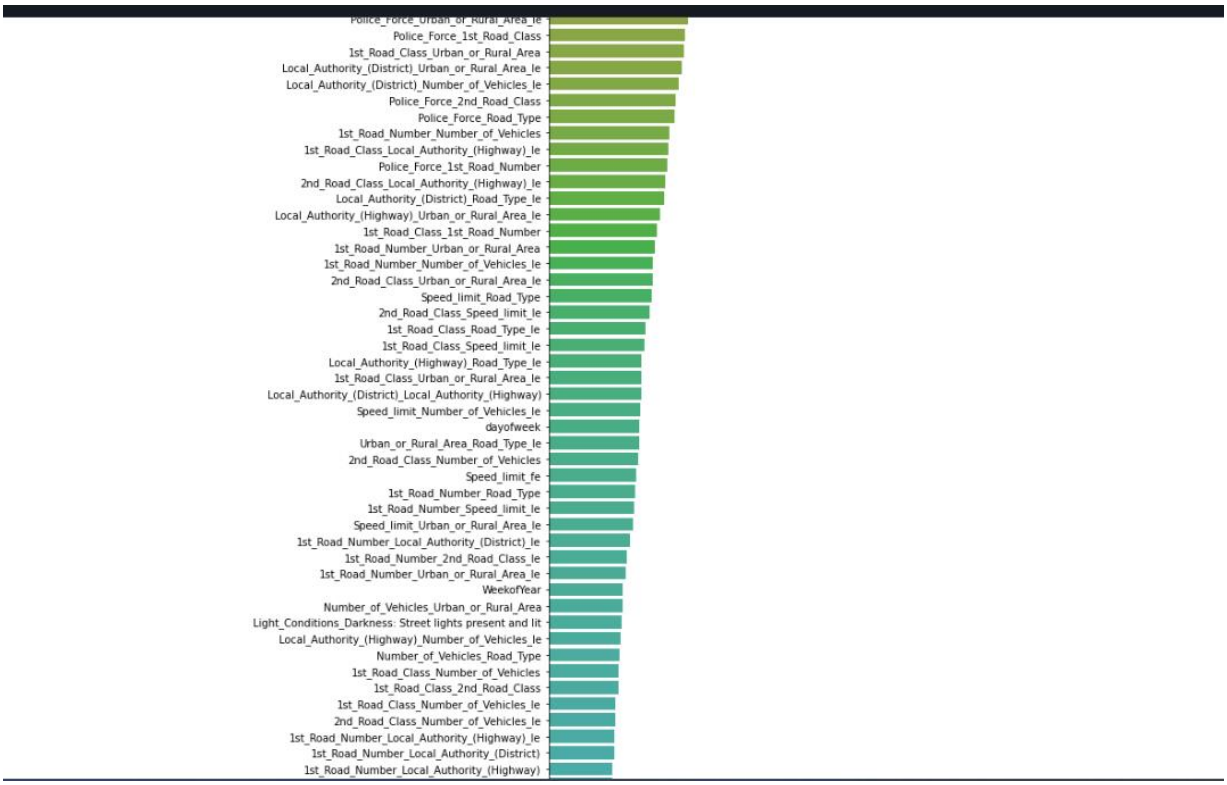
Fig 19: White box testing

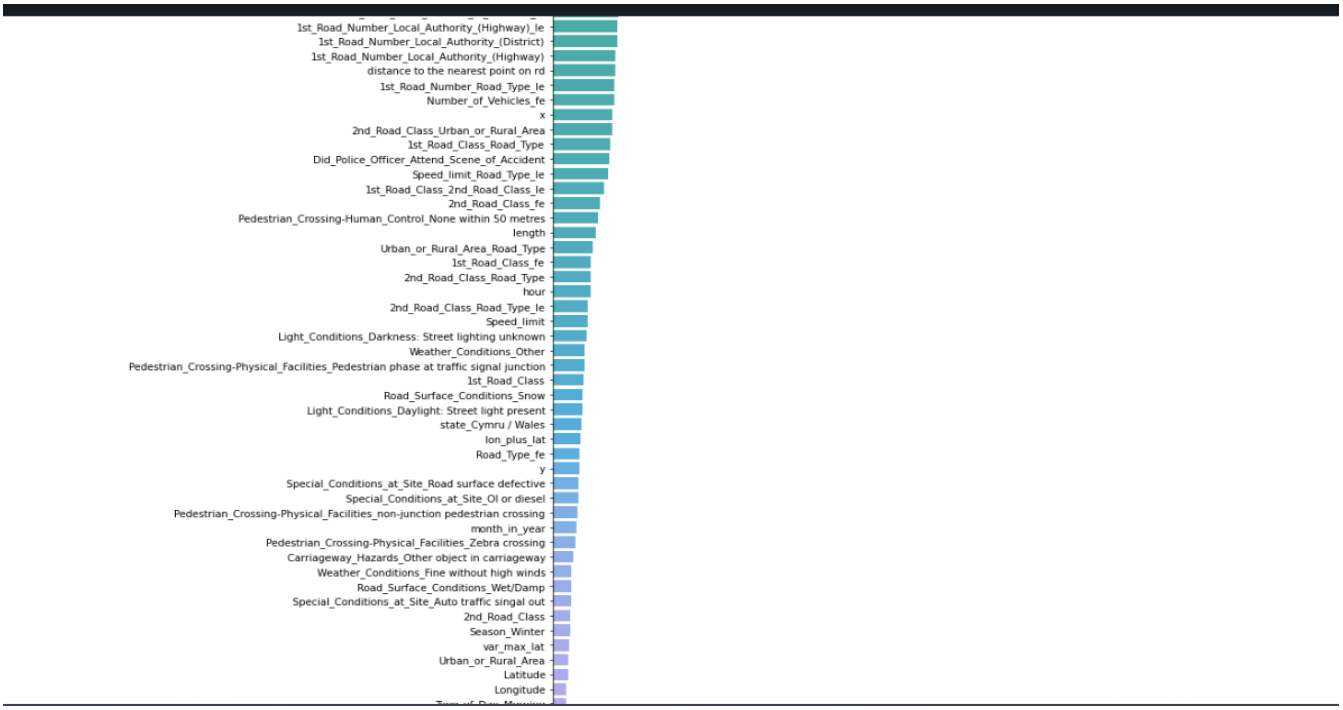
7.1.6 TESTING ON OUR SYSTEM:

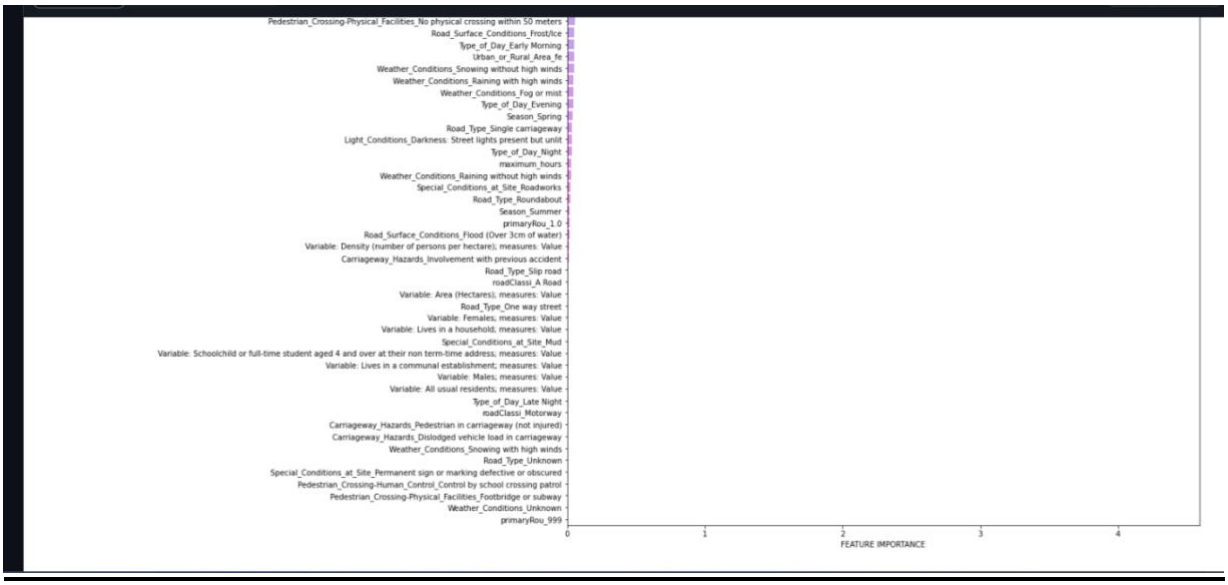
Testing a fully integrated software system is part of the system test. Software is usually integrated into a computer system (software itself is only a component of a computer system). In order to build a complete computer system, the software is constructed in modules and then interfaced with hardware and other applications. In other words, a computer system is a collection of software that performs various functions, but only software can do so because it has to communicate with appropriate hardware.

8.RESULTS


CATBOOST FEATURE IMPORTANCE
LAD_cnt
density_by_Police_Force
PF_cnt
Police_Force_Urban_or_Rural_Area
population_by_Police_Force
Police_Force_Speed_limit
postcode_cnt
Minute_Freq
1st_Road_Number
Number_of_Vehicles
2nd_Road_Class_Speed_limit
Police_Force_Local_Authority_(District)
Local_Authority_(District)_Number_of_Vehicles
Police_Force_Local_Authority_(Highway)
2nd_Road_Class_Local_Authority_(District)_le
Local_Authority_(Highway)_Urban_or_Rural_Area
Minute
Speed_limit_Local_Authority_(District)_le
Speed_limit_Number_of_Vehicles
1st_Road_Class_Local_Authority_(Highway)
Local_Authority_(Highway)_Road_Type
Police_Force_Speed_limit_le
Local_Authority_(District)_Urban_or_Rural_Area
Speed_limit_Local_Authority_(Highway)_le
Date
Police_Force_1st_Road_Class_le
Speed_limit_Local_Authority_(Highway)
1st_Road_Class_Speed_limit
1st_Road_Number_2nd_Road_Class
LAH_cnt
Speed_limit_Local_Authority_(District)
Police_Force_Road_Type_le
Police_Force_2nd_Road_Class_le
1st_Road_Class_Local_Authority_(District)_le
2nd_Road_Class_Local_Authority_(District)
Local_Authority_(District)_Local_Authority_(Highway)_le
1st_Road_Class_Local_Authority_(District)
Police_Force_1st_Road_Number_le
1st_Road_Number_fe
1st_Road_Number_Speed_limit
Local_Authority_(Highway)_Number_of_Vehicles
2nd_Road_Class_Local_Authority_(Highway)
Police_Force_Number_of_Vehicles_le
Police_Force_Local_Authority_(District)_le
1st_Road_Class_1st_Road_Number_le
Police_Force_Local_Authority_(Highway)_le
Number_of_Vehicles_Urban_or_Rural_Area_le
Number_of_Vehicles_Road_Type_le
day_in_month
Speed_limit_Urban_or_Rural_Area
Police_Force_Number_of_Vehicles
Local_Authority_(District)_Road_Type
Police_Force_Urban_or_Rural_Area_le
Police_Force_1st_Road_Class
1st_Road_Class_Urban_or_Rural_Area
Local_Authority_(District)_Urban_or_Rural_Area_le
Local_Authority_(District)_Number_of_Vehicles_le
Police_Force_2nd_Road_Class
Police_Force_Road_Type
1st_Road_Number_Number_of_Vehicles
1st_Road_Class_Local_Authority_(Highway)_le
Police_Force_1st_Road_Number
2nd_Road_Class_Local_Authority_(Highway)_le
Local_Authority_(District)_Road_Type_le
Local_Authority_(Highway)_Urban_or_Rural_Area_le
1st_Road_Class_1st_Road_Number
1st_Road_Number_Urban_or_Rural_Area
1st_Road_Number_Number_of_Vehicles_le
2nd_Road_Class_Urban_or_Rural_Area_le
Speed_limit_Road_Type
2nd_Road_Class_Speed_limit_le
1st_Road_Class_Road_Type_le
1st_Road_Class_Speed_limit_le
Local_Authority_(Highway)_Road_Type_le
1st_Road_Class_Urban_or_Rural_Area_le
Local_Authority_(District)_Local_Authority_(Highway)
Speed_limit_Number_of_Vehicles_le


Police_Force_Urban_or_Rural_Area_le
Police_Force_1st_Road_Class
1st_Road_Class_Urban_or_Rural_Area
Local_Authority_(District)_Urban_or_Rural_Area_le
Local_Authority_(District)_Number_of_Vehicles_le
Police_Force_2nd_Road_Class
Police_Force_Road_Type
1st_Road_Number_Number_of_Vehicles
1st_Road_Class_Local_Authority_(Highway)_le
Police_Force_1st_Road_Number
2nd_Road_Class_Local_Authority_(Highway)_le
Local_Authority_(District)_Road_Type_le
Local_Authority_(Highway)_Urban_or_Rural_Area_le
1st_Road_Class_1st_Road_Number
1st_Road_Number_Urban_or_Rural_Area
1st_Road_Number_Number_of_Vehicles_le
2nd_Road_Class_Urban_or_Rural_Area_le
Speed_limit_Road_Type
2nd_Road_Class_Speed_limit_le
1st_Road_Class_Road_Type_le
1st_Road_Class_Speed_limit_le
Local_Authority_(Highway)_Road_Type_le
1st_Road_Class_Urban_or_Rural_Area_le
Local_Authority_(District)_Local_Authority_(Highway)
Speed_limit_Number_of_Vehicles_le
dayofweek
Urban_or_Rural_Area_Road_Type_le
2nd_Road_Class_Number_of_Vehicles
Speed_limit_fe
1st_Road_Number_Road_Type
1st_Road_Number_Speed_limit_le
Speed_limit_Urban_or_Rural_Area_le
1st_Road_Number_Local_Authority_(District)_le
1st_Road_Number_2nd_Road_Class_le
1st_Road_Number_Urban_or_Rural_Area_le
WeekofYear
Number_of_Vehicles_Urban_or_Rural_Area
Light_Conditions_Darkness: Street lights present and lit
Local_Authority_(Highway)_Number_of_Vehicles_le
Number_of_Vehicles_Road_Type
1st_Road_Class_Number_of_Vehicles
1st_Road_Class_2nd_Road_Class
1st_Road_Class_Number_of_Vehicles_le
2nd_Road_Class_Number_of_Vehicles_le
1st_Road_Number_Local_Authority_(Highway)_le
1st_Road_Number_Local_Authority_(District)
1st_Road_Number_Local_Authority_(Highway)


1st_Road_Number_Local_Authority_(Highway)_le
1st_Road_Number_Local_Authority_(District)
1st_Road_Number_Local_Authority_(Highway)
distance to the nearest point on rd
1st_Road_Number_Road_Type_le
Number_of_Vehicles_fe
x
2nd_Road_Class_Urban_or_Rural_Area
1st_Road_Class_Road_Type
Did_Police_Officer_Attend_Scene_of_Accident
Speed_limit_Road_Type_le
1st_Road_Class_2nd_Road_Class_le
2nd_Road_Class_fe
Pedestrian_Crossing-Human_Control_None within 50 metres
length
Urban_or_Rural_Area_Road_Type
1st_Road_Class_fe
2nd_Road_Class_Road_Type
hour
2nd_Road_Class_Road_Type_le
Speed_limit
Light_Conditions_Darkness: Street lighting unknown
Weather_Conditions_Other
Pedestrian_Crossing-Physical_Facilities_Pedestrian phase at traffic signal junction
1st_Road_Class
Road_Surface_Conditions_Snow
Light_Conditions_Daylight: Street light present
state_Cymru / Wales
lon_plus_lat
Road_Type_fe
y
Special_Conditions_at_Site_Road surface defective
Special_Conditions_at_Site_Oil or diesel
Pedestrian_Crossing-Physical_Facilities_non-junction pedestrian crossing
month_in_year
Pedestrian_Crossing-Physical_Facilities_Zebra crossing
Carriageway_Hazards_Other object in carriageway
Weather_Conditions_Fine without high winds
Road_Surface_Conditions_Wet/Damp
Special_Conditions_at_Site_Auto traffic singal out
2nd_Road_Class
Season_Winter
var_max_lat
Urban_or_Rural_Area
Latitude
Longitude




```
((49772, 2),
           postcode  Accident_risk_index
 postcode
 AB10 1AU  AB10 1AU             1.344906
 AB10 1PG  AB10 1PG             1.357971
 AB10 1TT  AB10 1TT             1.477456
 AB10 1YP  AB10 1YP             1.388434
 AB10 6LQ  AB10 6LQ             1.344694
 AB10 6NN  AB10 6NN             1.553909
 AB10 7FT  AB10 7FT             1.452170
 AB10 7JP  AB10 7JP             1.238208
 AB10 7LY  AB10 7LY             1.210661
 AB11 5BD  AB11 5BD             1.320834)
```

Fig 20:Risk Score Prediction

9.CONCLUSION AND FUTURE ENCHANCEMENT

9.1 CONCLUSION:

In conclusion, the application of machine learning algorithms for accident risk score prediction represents a promising and impactful approach in enhancing safety measures across various domains. Through the utilization of historical data and advanced predictive models, machine learning has demonstrated its capability to assess and forecast potential risks with a considerable degree of accuracy.

The development and implementation of these predictive models enable stakeholders to proactively identify high-risk areas, patterns, and contributing factors associated with accidents. This empowers decision-makers to allocate resources efficiently, implement targeted interventions, and devise preemptive strategies to mitigate risks and prevent potential accidents.

However, it's crucial to acknowledge that while machine learning offers valuable insights, its effectiveness is contingent upon the quality and relevance of the data used for training and validation. Additionally, ethical considerations regarding data privacy, fairness, and transparency must be rigorously addressed to ensure responsible and unbiased utilization of predictive algorithms.

Continued research, refinement of algorithms, and collaboration between domain experts and data scientists are essential for further improving the accuracy and reliability of accident risk score predictions. By leveraging the potential of machine learning, we can aspire to create safer environments, reduce accident rates, and ultimately save lives.

Accident risk prediction through the lens of machine learning represents a profound stride in the realm of safety management and risk mitigation. Harnessing the power of data-driven insights, predictive models crafted by machine learning algorithms have shown tremendous potential in revolutionizing how we perceive, analyze, and proactively address potential hazards and accidents across various industries and domains.

The amalgamation of historical incident data, coupled with the prowess of machine learning algorithms, enables the creation of predictive models capable of discerning intricate patterns, correlations, and risk factors that might have previously eluded human observation. By sifting through vast datasets, these algorithms learn to identify subtle nuances and relationships that contribute to accident occurrences, empowering them to predict potential risks with a commendable level of accuracy.

The significance of this predictive capability is monumental. Not only does it allow for the identification of high-risk zones and scenarios, but it also enables stakeholders and decision-

makers to allocate resources efficiently and implement targeted interventions before accidents transpire. Whether it's in transportation, manufacturing, healthcare, or any other sector prone to accidents, these predictive models serve as a pivotal tool for preemptive planning, helping to avert potential disasters and minimize their impact on lives and resources.

However, the effectiveness of machine learning algorithms in predicting accident risks is inherently tied to the quality, relevance, and diversity of the data used in their training and validation. Moreover, ethical considerations surrounding data privacy, fairness, and transparency need to be rigorously addressed to ensure responsible deployment and utilization of these predictive systems. Bias in data or algorithms could lead to skewed predictions, exacerbating existing disparities or overlooking certain risk factors, thereby undermining the effectiveness of the predictions.

Continued collaboration between domain experts, data scientists, and policymakers is imperative to further refine these predictive models. Enhanced accuracy, interpretability, and generalizability are areas that demand continuous research and development efforts. Moreover, fostering a culture of shared learning, where insights from different sectors and interdisciplinary perspectives converge, can significantly bolster the effectiveness and applicability of these predictive models.

The ultimate goal of leveraging machine learning for accident risk prediction transcends mere statistical forecasts; it aims to create safer environments, prevent accidents, and, most importantly, safeguard lives. As these predictive models continue to evolve and improve, they hold the promise of revolutionizing safety protocols, shaping policies, and fundamentally altering the way we approach risk management across diverse industries, fostering a future where accidents are minimized, if not entirely prevented.

9.2 FUTURE ENCHANCEMENT:

The future trajectory of accident risk score prediction systems is poised for remarkable advancements, envisaging a holistic evolution aimed at refining the accuracy, depth, and practical applicability of predictive models. A pivotal avenue for progress lies in the integration of an array of diverse, real-time data sources. This integration, spanning from IoT devices to social media feeds, offers the promise of a more comprehensive understanding of the multifaceted contextual factors that intricately contribute to the occurrence and likelihood of accidents. Real-time data streams have the potential to provide immediate insights, enabling a proactive approach to risk assessment and mitigation.

Moreover, the continual refinement of feature engineering techniques and the exploration of more sophisticated model architectures, such as deep learning paradigms, present an opportunity to capture the intricate and often subtle relationships embedded within datasets. By delving deeper into data patterns, these advancements aim to significantly enhance the predictive accuracy of these models, enabling them to discern nuanced risk factors that might have previously evaded detection.

In parallel, the development of models with enhanced explainability and transparency—commonly referred to as Explainable AI (XAI)—stands as an imperative stride towards fostering trust and understanding among stakeholders. These explainable models illuminate the decision-making process, enabling users to comprehend how predictions are formulated and strengthening the usability and acceptance of these predictive systems.

Furthermore, the progression towards adaptive learning mechanisms that continually assimilate and adapt to new data in real-time is crucial. This adaptability ensures that predictive models remain updated and efficacious in dynamic and evolving environments. Collaboration between data scientists and domain experts remains pivotal, injecting domain-specific knowledge and expertise into the model refinement process. This collaborative approach ensures that predictive models account for context-specific factors, augmenting their relevance and practicality in real-world scenarios.

Ethical considerations, bias mitigation, and the pursuit of fairness in predictive models stand as ongoing imperatives. Efforts to address biases in both data and algorithms are essential, ensuring equitable and unbiased predictions, especially in high-stakes applications such as accident risk prediction.

Encouraging cross-sector collaboration for data sharing and fostering the development of user-friendly interfaces that facilitate broader accessibility are crucial steps in broadening the reach and impact of these predictive systems across diverse industries and sectors.

Embracing these multifaceted advancements in accident risk score prediction underscores a comprehensive and multidisciplinary approach. This approach amalgamates technological innovations, ethical considerations, collaborative efforts, and a focus on practical applicability. By assimilating these advancements, the potential to substantially elevate the efficacy and societal impact of accident risk prediction systems is palpable, fostering a safer and more proactive approach to risk management across various domains.

10.BIBILOGRAPHY

[1] https://www.ijraset.com/research-paper/road-accident-prediction-model-using-ml

[2]https://ieeexplore.ieee.org/document/9823499

[3]https://www.irjmets.com/uploadedfiles/paper/issue_5_may_2022/23616/final/fin_irjmets1653072961.pdf

[4]https://www.mdpi.com/2071-1050/15/7/5939

[5]https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4460883

[6] "Predictive Modeling and Risk Assessment",Author: A. Gailus,Springer, 2017

[7] https://link.springer.com/article/10.1007/s42452-020-3125-1

[8]https://medium.com/geoai/using-machine-learning-to-predict-car-accident-risk-4d92c91a7d57

[9] J. Milton, V. Shankar, F. Mannering (2008) "Highway accident severities and the mixed logit model: an exploratory empirical analysis", Accident Analysis & Prevention, Volume 40, pp. 260-266

[10] Journal of the Transportation Research Board, Volume 2432, pp 91-98

[11] Chao Wang, Mohammed A.Quddus, Stephen G.Ison (2011) "Predicting accident frequency at their severity levels and its application in site ranking using a two-stage mixed multivariate model", Accident Analysis & Prevention, Volume 43,issue 6, pp. 1979- 1990, DOI: 10.1016/j.aap.2011.05.016

[12] Maryam Dastoorpoor, Esmaeil Idani, Narges Khanjani, Gholamreza Goudarzi, Abbas Bahrampour (2016) "Relationshipbetween air pollution, weather, traffic, and trafficrelated mortality", Trauma Mon, Volume 21, issue 4, pp. e37585. PMID:2818

[13] Athanasios Theofilatos, George Yannis, Pantelis Kopelias, Fanis Papadimitriou (2016) "Predicting Road accidents: a rare-events modeling approach, Transportation Research Procedia", Volume 14, pp. 3399- 3405

[14] Adanu, E.K.; Hainen, A.; Jones, S. Latent class analysis of factors that influence weekday and weekend single-vehicle crash severities. *Accid. Anal. Prev.* **2018**, *113*.

[15] Novikov, A.; Novikov, I.; Shevtsova, A. Study of the impact of type and condition of the road surface on parameters of signalized intersection. *Transp. Res. Procedia* **2018**, *36*, 548–555.

[16]https://www.researchgate.net/publication/359975404_Traffic_Accident_Risk_Prediction_Using_Machine_Learning

[17] https://www.frontiersin.org/articles/10.3389/fbuil.2022.860805/full

[18]researchgate.net/publication/354589124_RFCNN_Traffic_Accident_Severity_Prediction_Based_on_Decision_Level_Fusion_of_Machine_and_Deep_Learning_Model

[19]https://digitalcommons.harrisburgu.edu/cgi/viewcontent.cgi?article=1001&context=anms_dandt

[20] https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4460883

[21]https://datasci.columbian.gwu.edu/sites/g/files/zaxdzs4746/files/downloads/Capstones/Capstone%20%20Final%20Report%20%20Cristina%20Giraldo%20and%20Taisha%20Ferguson%20%282%29.pdf