**B.Tech. BCSE497J - Project-I**

**OPTIMISING BUSINESS LICENCE ISSUANCE USING PREDICTIVE ANALYSIS**

*Submitted in partial fulfillment of the requirements for the degree of*

**Bachelor of Technology**

*in*

**Programme**

*by*

|  |  |
| --- | --- |
| **21BCT0140** | **P.V.SATWIK KRISHNA** |
| **21BCI0356** | **Dereddy Maheswarareddy** |
| **21BCE3245** | **Kanala Chanikya Reddy** |

**Under the Supervision of**

|  |
| --- |
| **Dr. KOVENDAN AKP** |
| Assistant Professor Senior Grade 1 |
| School of Computer Science and Engineering (SCOPE) |



November 2024

**DECLARATION**

I hereby declare that the project entitled OPTIMIZING BUSINESS LICENSE ISSUANCE USING PREDICTIVE ANALYSIS submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Prof. / Dr. KOVENDAN A K P

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore Date: 13/11/2024

**Signature of the Candidate**

**CERTIFICATE**

This is to certify that the project entitled OPTIMIZING BUSINESS LICENSE ISSUANCE USING PREDICTIVE ANALYSIS submitted by **P V SATWIK KRISHNA**, DEREDDY MAHESWARA REDDY, KANALA CHANIKYA REDDY **School of Computer Science and Engineering**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him / her under my supervision during Fall Semester 2024-2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The project fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 13-11-2024

**Signature of the Guide**

**Examiner(s)**

DR. SHAMILA BHANU

COMPUTER SCIENCE – INTERNET OF THINGS

**ACKNOWLEDGEMENTS**

I am deeply grateful to the management of Vellore Institute of Technology (VIT) for providing me with the opportunity and resources to undertake this project. Their commitment to fostering a conducive learning environment has been instrumental in my academic journey. The support and infrastructure provided by VIT have enabled me to explore and develop my ideas to their fullest potential.

My sincere thanks to Dr. Ramesh Babu K, the Dean of the School of Computer Science and Engineering (SCOPE), for his unwavering support and encouragement. His leadership and vision have greatly inspired me to strive for excellence. The Dean’s dedication to academic excellence and innovation has been a constant source of motivation for me. I appreciate his efforts in creating an environment that nurtures creativity and critical thinking.

I express my profound appreciation to DR. SHARMILA BHANU, the Head of the Computer Science – Internet of things department, for his/her insightful guidance and continuous support. His/her expertise and advice have been crucial in shaping the direction of my project. The Head of Department’s commitment to fostering a collaborative and supportive atmosphere has greatly enhanced my learning experience. His/her constructive feedback and encouragement have been invaluable in overcoming challenges and achieving my project goals.

I am immensely thankful to my project supervisor, KOVENDAN A K P, for his/her dedicated mentorship and invaluable feedback. His/her patience, knowledge, and encouragement have been pivotal in the successful completion of this project. My supervisor’s willingness to share his/her expertise and provide thoughtful guidance has been instrumental in refining my ideas and methodologies. His/her support has not only contributed to the success of this project but has also enriched my overall academic experience.

Thank you all for your contributions and support.

**P V SATWIK KRISHNA**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Contents** | **Page No.** |
|  | **ABSTRACT** |  |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 Background | **1** |
|  | 1.2 Motivations | **2** |
|  | 1.3 Scope of the Project | **3** |
| **2.** | **PROJECT DESCRIPTION AND GOALS** |  |
|  | 2.1 Literature Review | **4** |
|  | 2.2 Research Gap | **4** |
|  | 2.3 Objectives | **4** |
|  | 2.4 Problem Statement | **5** |
|  | 2.5 Project Plan | **5** |
| **3.** | **TECHNICAL SPECIFICATION** |  |
|  | 3.1 Requirements |  |
|  | 3.1.1 Functional | **6** |
|  | 3.1.2 Non-Functional | **7** |
|  | 3.2 Feasibility Study |  |
|  | 3.2.1 Technical Feasibility | **8** |
|  | 3.2.2 Economic Feasibility | **9** |
|  | 3.2.2 Social Feasibility | **10** |
|  | 3.3 System Specification |  |
|  | 3.3.1 Hardware Specification | **11** |
|  | 3.3.2 Software Specification | **12** |
| **4.** | **DESIGN APPROACH AND DETAILS** |  |
|  | 4.1 System Architecture | **14** |
|  | 4.2 Design |  |
|  | 4.2.1 Data Flow Diagram | **15** |
|  | 4.2.2 Use Case Diagram | **17** |
|  | 4.2.3 Class Diagram | **19** |
|  | 4.2.4 Sequence Diagram | **21** |
| **5.** | **METHODOLOGY AND TESTING** |  |
|  | **5.1 Module Description** | **25** |
|  | **5.2 Testing** | **26** |
| **6.** | **PROJECT DEMONSTRATION** | **29** |
| **7.** | **RESULT AND DISCUSSION** | **32** |
| **8.** | **CONCLUSION** | **34** |
| **9.** | **REFERENCES** | **35** |
| **10** | **APPENDIX A – SAMPLE CODE** | **36** |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| 1. | System Architecture | 14 |
|  |  |  |
| 2.  3.  4.  5.    6.  7.  8.  9. | Data Flow Diagram    Use Case Diagram  Class Diagram  Sequence Digram  Gnatt Chart  Missing Values graph  Bar graph for average days  Bar graph for for distribution | 15  17  19  21  23  29  29  30 |

**ABSTRACT**

The inefficiencies in the business license issuance process have led to delays and dissatisfaction among stakeholders, as well as revenue losses for businesses and governments. This project explores the integration of Machine Learning (ML) and predictive analytics to optimize the business license issuance process, focusing on its ability to analyze historical data, identify bottlenecks, and predict the time required to issue a license. By leveraging ML algorithms such as Linear Regression, Lasso Regression, and Ridge Regression, this project aims to reduce the average time to issue a license and streamline operations.

The use of supervised learning techniques enables the system to detect inefficiencies in the application and approval stages, while predictive models provide real-time estimates for license issuance timelines. The project also examines the challenges in deploying predictive models for regulatory processes, including data availability, model accuracy, and external factors influencing delays. Through data analysis and model testing, the project demonstrates how ML can be integrated into the current licensing framework to increase efficiency, reduce wait times, and improve overall service quality.

To further enhance the effectiveness of the ML-based system, it is crucial to address data quality and preprocessing, model selection and evaluation, deployment and monitoring, and ethical considerations. By meticulously cleaning and preparing the data, selecting the most appropriate ML algorithms, and continuously monitoring and refining the model, governments can maximize the benefits of this technology. Additionally, it is imperative to prioritize fairness, transparency, and privacy to ensure the ethical and responsible use of ML in this context..

This research highlights the potential for ML-based systems to transform administrative processes, emphasizing the importance of continued innovation in this area to support economic growth and business development.

**Keywords**: Predictive Analytics, Machine Learning, License Issuance, Lasso Regression, Data Analysis.

**1. INTRODUCTION**

**1.1 Background**

The increasing complexity of regulatory processes and the growing number of business applications have led to significant delays in the issuance of business licenses, affecting both entrepreneurs and local governments. Traditional methods for managing the application, review, and approval processes rely heavily on manual labor and outdated systems, resulting in inefficiencies, bottlenecks, and inconsistencies. These issues not only impact business operations but also lead to revenue losses and dissatisfaction among stakeholders.

In response to these challenges, Machine Learning (ML) and Predictive Analytics have emerged as transformative tools for optimizing administrative processes. By analyzing historical data and identifying patterns, ML algorithms can predict the time required to issue a business license, allowing regulatory bodies to streamline the process and allocate resources more efficiently. These technologies can help mitigate delays by providing real-time insights and forecasts, thus enabling proactive management of the licensing workflow.

This project aims to explore the potential of ML and Predictive Analytics to enhance the business license issuance process, focusing on their applications, benefits, and challenges. Through the development and testing of ML models, the project seeks to reduce inefficiencies, improve processing times, and create a more responsive regulatory system that adapts to evolving business needs.

**1.2 Motivation**

The ever-growing number of business applications, coupled with the complexity of regulatory compliance, has made traditional methods of issuing licenses increasingly inadequate. Conventional approaches, which rely on manual processes, static workflows, and outdated systems, are often too slow and inefficient to handle the rising demand for timely business licensing. As local governments and businesses become more dependent on efficient administrative processes, the stakes for improving the licensing workflow have never been higher. Delays in license issuance can lead to significant consequences, including lost revenue, business disruptions, and dissatisfaction among both business owners and regulatory bodies.

This project is motivated by the potential of **Machine Learning (ML)** and **Predictive Analytics** to transform the business license issuance process. Unlike traditional manual methods, ML algorithms can analyze historical data, detect patterns in processing times, and predict future delays, allowing for a more proactive and adaptive system. These technologies can identify bottlenecks, streamline workflows, and provide real-time insights that enable regulatory agencies to better manage resources and reduce the overall time required for license issuance.

The goal of this project is to explore how ML and predictive models can be integrated into existing regulatory frameworks, providing a smarter and more efficient approach to business licensing. By addressing the limitations of current processes, this project aims to improve the speed, accuracy, and reliability of license issuance, contributing to smoother business operations and economic growth.

**1.3 Scope of the Project**

The scope of this project focuses on exploring the transformative potential of **Machine Learning (ML)** and **Predictive Analytics** in enhancing the business license issuance process. The project will begin by identifying and analyzing the current challenges faced by regulatory bodies, including inefficiencies, bottlenecks, and unpredictable processing times. By understanding the limitations of traditional manual methods, the project aims to highlight areas where ML and predictive models can provide significant improvements in the speed and accuracy of license issuance.

The project will involve the development and implementation of ML models for tasks such as processing time prediction, bottleneck detection, and workflow optimization. Additionally, the project will explore various predictive models like **Linear Regression**, **Lasso Regression**, and **Ridge Regression** to forecast the time required for license approval based on historical data. The effectiveness of these models will be tested on real-world datasets to ensure their reliability in predicting license issuance times and improving the overall process efficiency.

The project also considers the practical and operational challenges of integrating ML into existing regulatory frameworks. This includes addressing concerns about data quality, model accuracy, and scalability. Ultimately, the project aims to contribute to the development of a more efficient and adaptive business license issuance system that can streamline operations and better serve businesses and governments alike.

**2. PROJECT DESCRIPTION AND GOALS**

**2.1 Literature Review**

The literature on the application of Machine Learning (ML) and predictive analytics in administrative processes is growing. Several studies have highlighted the potential of ML algorithms in predicting processing times and identifying inefficiencies in government workflows. The use of **Linear Regression**, **Lasso Regression**, and **Ridge Regression** in various industries has demonstrated their ability to handle time-series data and predict outcomes based on historical patterns. Studies such as those by [Author et al., Year] explore the use of ML in administrative reforms, including optimizing the allocation of resources for public services.

In the business license context, research on the application of ML has been limited, though related work in process automation and workflow optimization suggests significant potential. Key findings emphasize the need for real-time predictions and the integration of historical and operational data to reduce the average time for license issuance. These studies provide a foundation for exploring how predictive models can be tailored to specific regulatory processes, but they lack the empirical focus on business licensing workflows.

**2.2 Research Gap**

While substantial research exists on applying ML in operational and process optimization, there is a notable gap in how these technologies are applied specifically to business license issuance. Existing studies focus on general administrative efficiency, but few address the specific challenges posed by complex regulatory frameworks in government agencies. Furthermore, research lacks insights into how **predictive analytics** can address bottlenecks such as documentation reviews and compliance checks within the business licensing process. This project aims to bridge that gap by providing an in-depth analysis of how ML models can be applied to optimize the entire business license issuance pipeline.

**2.3 Objectives**

The main objective of this project is to develop an efficient predictive system that leverages **Machine Learning (ML)** models to forecast the time required for issuing business licenses, thus streamlining the entire license issuance process. By analyzing historical data and identifying patterns in the workflow, the project aims to reduce inefficiencies and delays that often hinder the approval process. Specifically, this project will utilize models such as **Linear Regression**, **Lasso Regression**, and **Ridge Regression** to predict the time frames for various stages of the licensing process, including application reviews, compliance checks, and final approvals. Additionally, the project seeks to identify bottlenecks in the workflow and provide actionable insights for resource optimization, enabling regulatory agencies to allocate their resources more effectively and reduce waiting times. The ultimate goal is to create a smarter, data-driven solution that will enhance the efficiency of regulatory bodies, improve service quality, and support smoother business operations, all while maintaining accuracy and compliance

**2.4 Problem** **Statement**

The current business license issuance process suffers from inefficiencies and delays, negatively impacting both regulatory bodies and businesses. With manual processing methods and outdated systems, regulatory agencies struggle to handle the increasing volume of applications in a timely manner. These delays lead to frustration among business owners, loss of revenue, and reduced operational efficiency. The problem is further exacerbated by bottlenecks in compliance verification, document reviews, and approval workflows. Therefore, there is a need for a predictive system that can streamline the process, provide accurate forecasts for license issuance times, and help regulatory bodies better allocate resources and reduce delays.

**2.5** **Project Plan**

The project will follow a structured plan divided into several key phases to ensure the successful development and implementation of the predictive analytics system. In the initial phase, spanning the first two months, the project will focus on collecting and preprocessing historical business license issuance data from the City of Chicago’s data portal. This stage involves cleaning the data, handling missing values, and encoding categorical variables to prepare the dataset for analysis. Once the data is prepared, the next phase, scheduled for months three and four, will involve developing and training **Machine Learning (ML)** models, such as **Linear Regression**, **Lasso Regression**, and **Ridge Regression**, to predict the time required for license issuance. The accuracy of these models will be validated using cross-validation techniques.

**3. TECHNICAL SPECIFICATION**

**3.1 Requirements**

**3.1.1 Functional**

**Data Collection and Storage**:  
The system must be able to gather historical data on business license issuance from sources such as the City of Chicago’s Business Affairs and Consumer Protection portal. This data includes information like application dates, compliance checks, approval timelines, and issuance dates.

**Data Preprocessing**:  
The system should preprocess the collected data by handling missing values, encoding categorical variables (e.g., types of licenses), and normalizing numerical values (e.g., average and median days to issue licenses).

**Machine Learning Model Development**:  
The system must develop and train multiple **Machine Learning (ML)** models such as **Linear Regression**, **Lasso Regression**, and **Ridge Regression** to predict license issuance times.

**Prediction and Analysis**:  
The system should provide real-time predictions for the time required to issue licenses based on input data. It must also detect bottlenecks in the licensing process, such as delays in compliance checks or documentation reviews.

**Evaluation and Testing**:  
The system must allow for performance evaluation through cross-validation and other model validation techniques. Model performance metrics such as **Mean Absolute Error (MAE)** and **R² Score** should be tracked to ensure reliability.

**3.1.2 Non-Functional**

**Performance**:  
The system must be designed to provide real-time predictions with minimal latency. Since the goal is to optimize the business license issuance process, the system must handle data processing and model predictions quickly, ensuring that regulatory authorities receive timely insights. Response times should be optimized so that predictions on license issuance times are generated within seconds after new application data is submitted. The system should also be able to process large volumes of data without significant slowdowns.

**Scalability**:  
As the volume of business license applications grows, the system must scale to handle increasing datasets without degradation in performance. This includes the ability to process additional historical data as well as real-time data inputs, and potentially more complex data from additional cities or regulatory bodies. The architecture should support horizontal scaling, allowing the system to add more processing power as data needs grow.

**Reliability**:  
The system must be highly reliable, ensuring consistent uptime and availability. It should be capable of delivering accurate and reliable predictions even under high load conditions. Failures in data collection, model prediction, or user interface components should be handled gracefully, with proper logging and error reporting, and automatic recovery mechanisms in place to minimize downtime.

**Security**:  
Given that sensitive data, such as business applicant details and compliance information, will be processed, the system must have strong security protocols in place. All data must be encrypted both at rest and in transit. Access to the system should be controlled via user authentication and role-based access control (RBAC) to ensure that only authorized personnel can access specific data and functionalities.

**Usability**:  
The system should be designed with the end-users in mind, specifically regulatory staff who may not have technical expertise. The user interface (UI) should be intuitive and easy to navigate, with clear visualizations and options for filtering and sorting data.

**3.2 Feasibility Study**

**3.2.1 Technical Feasibility**

Technology Availability:  
The project leverages existing Machine Learning (ML) and Predictive Analytics technologies, which are well-documented and widely used in various domains including administrative process optimization. Tools such as scikit-learn, TensorFlow, and PyTorch offer a robust ecosystem for building, training, and deploying predictive models. Data processing tools like Pandas and NumPy are readily available and highly efficient for working with structured data.

Technical Expertise:  
The project requires a team with expertise in data science, machine learning, and predictive analytics. Specifically, knowledge of handling structured datasets, building regression models, and analyzing workflow bottlenecks is crucial. The team should also be proficient in using ML frameworks such as scikit-learn and TensorFlow, as well as cloud platforms like AWS or Google Cloud for scalable solutions. If necessary, training can be provided to enhance the team's knowledge of specific ML techniques or data pipeline management.

Infrastructure:  
Adequate computational resources, such as high-performance servers or cloud-based computing solutions, are necessary to process the large volumes of historical business license data. Cloud platforms such as Google Cloud Platform (GCP) or Amazon Web Services (AWS) provide scalable infrastructure, allowing for flexibility in storage and computational power. Access to such infrastructure will ensure that the models can be trained efficiently and predictions can be made in real-time, even as data grows.

Integration:  
The system must integrate seamlessly with existing government or regulatory IT infrastructure. This includes integrating with databases or systems used for storing application data, compliance checks, and other regulatory processes. Additionally, the system should be able to communicate with existing tools used for tracking and managing business licenses to ensure smooth data flow and real-time updates. Integration with external APIs for data fetching and validation should also be considered to streamline the workflow.

D**ata Security and Privacy Compliance**:  
Given the sensitive nature of government and business data, it's crucial to comply with data protection regulations such as GDPR or CCPA. The system should incorporate security protocols such as data encryption (both in transit and at rest), access control, and regular audits. Role-based access should be enforced to prevent unauthorized access to data and models, ensuring only qualified personnel can interact with sensitive data.

**3.2.2 Economic Feasibility**

Initial Investment:  
The primary costs for the project will involve developing the machine learning models, setting up infrastructure, and integrating the system into the existing business license issuance workflow. These costs will cover:

* Data collection: Accessing and cleaning historical data from public or internal sources.
* Model development: Developing, training, and validating machine learning models to predict business license issuance times.
* Infrastructure setup: Using cloud platforms like Google Cloud Platform (GCP) or Amazon Web Services (AWS) to handle data storage and processing requirements. Initial costs will include setting up secure and scalable cloud computing resources.
* Personnel costs: Salaries for the data science team responsible for model development, system integration, and maintenance. This includes both internal employees and potential contractors or consultants.

**3.2.3 Social Feasibility**

User Acceptance:  
The key users of this system will be regulatory authorities, government staff, and business owners. The primary goal of the project is to improve the efficiency of the business license issuance process, which is currently slow and inefficient. A streamlined system will make the process more transparent and predictable, which will likely increase user satisfaction among both business owners and administrative personnel. A user-friendly interface with clear visualizations and automated reports will enable non-technical users to access and understand the system easily, improving its adoption. The system will also reduce the workload for administrative staff, allowing them to focus on more complex and high-priority tasks.

Impact on Workforce:  
While the system automates certain tasks involved in the business license issuance process, it will not entirely replace the need for human oversight. Instead, it will assist regulatory staff by automating routine tasks like tracking application progress and predicting timelines. This will enable staff to focus on more value-added activities, such as resolving exceptions or handling complex applications that require human intervention. Moreover, the system will likely reduce the need for frequent follow-ups or status inquiries from applicants, improving overall efficiency. The workforce will need to adapt to using data-driven decision-making tools, but this can be managed with adequate training and support.

Expected Organizational and Economic Impact:

The implementation of this system will have long-term benefits for the efficiency and effectiveness of the business license issuance process. Reduced processing times and greater transparency will likely lead to increased trust between regulatory bodies and business communities, enhancing the reputation of government agencies involved in business regulation. Additionally, by reducing operational delays, the system is expected to foster a more favorable environment for business growth and economic development, as it will allow entrepreneurs to launch their businesses more quickly.

**3.2 System Specification**

**3.2.1 Hardware Specification**

Processor:

A multi-core processor, preferably from the Intel Core i7/i9 or AMD Ryzen 7/9 series, is recommended to handle the data preprocessing, model training, and real-time predictions efficiently. For large datasets or complex models, high-performance servers with Intel Xeon or AMD EPYC processors may be required.

Memory (RAM):

At least 16 GB of RAM is recommended for initial development, especially for handling moderate datasets. For large-scale model training and data processing, 32 GB or more is preferred. Cloud solutions can also offer scalable memory options based on the dataset size.

Storage:

A minimum of 500 GB SSD storage is required to store datasets, model checkpoints, and other development files. For large datasets, cloud-based storage solutions like AWS S3 or Google Cloud Storage can be integrated to offer scalable storage.

Graphics Processing Unit (GPU):

For the training of more complex machine learning models (if needed), a dedicated GPU like NVIDIA Tesla or RTX series may be required to accelerate model training, especially for deep learning models. However, for standard regression models, a GPU is not strictly necessary. Cloud-based GPU solutions like AWS EC2 P3 Instances or Google Cloud TPUs can also be considered.

.

***3.2.2 Software Specification***

**Operating System**:  
The project can be developed on either **Windows 10/11**, **macOS**, or **Linux** (preferably **Ubuntu**). Linux-based environments are typically preferred for machine learning projects due to better support for open-source libraries and ease of deployment in cloud environments.

**Programming Languages**:

* **Python**: Python is the primary programming language for machine learning and predictive analytics. Its extensive library ecosystem and ease of use make it ideal for building and deploying machine learning models.
* **SQL**: SQL will be used to query, manage, and retrieve data from relational databases.

**Development Environment**:

* **Jupyter Notebooks** or **JupyterLab**: For interactive data analysis, model development, and experimentation.
* **PyCharm** or **VS Code**: These integrated development environments (IDEs) can be used for larger-scale software development and debugging.

**Libraries and Frameworks**:

* **scikit-learn**: For building and deploying machine learning models such as **Linear Regression**, **Lasso Regression**, and **Random Forest**.
* **Pandas** and **NumPy**: For data preprocessing, manipulation, and analysis.
* **Matplotlib** and **Seaborn**: For data visualization and plotting.
* **TensorFlow** or **PyTorch**: If deep learning models are needed in future iterations of the project.
* **XGBoost** or **LightGBM**: For advanced predictive modeling if required for more complex predictions.

**Database**:

* **MySQL** or **PostgreSQL**: Relational databases can be used to store historical business license data and model outputs. These databases will support large datasets and allow for querying and storing structured data.
* **AWS RDS** or **Google Cloud SQL**: Cloud-based database solutions can also be used for scalability and ease of access.

**Security Tools**:

* **SSL/TLS Encryption**: For securing data transmission, especially when dealing with sensitive business and applicant data.
* **AWS IAM** (Identity and Access Management) or **Google Cloud IAM**: For managing access control to the cloud infrastructure and ensuring that only authorized users can interact with the system.
* **Data Anonymization Tools**: To protect sensitive personal data, especially when using real-world business license applicant data, anonymization or pseudonymization techniques should be employed.
* **Version Control System**:  
  Utilizing Git and a platform like GitHub, GitLab, or Bitbucket is essential for version control, especially for collaboration and tracking model iterations. It allows multiple team members to work on code, experiment with different model versions, and document changes systematically.

**CI/CD Pipelines**:  
Implementing continuous integration and continuous deployment (CI/CD) pipelines using tools like Jenkins, GitHub Actions, or GitLab CI/CD will automate testing, integration, and deployment. This ensures that updates to the predictive model or system components are rigorously tested and deployed quickly.

**Containerization and Orchestration**:  
Using Docker for containerization ensures consistency across different environments and simplifies deployment. Kubernetes can be introduced to manage containers at scale, making it easier to handle peak loads and ensuring high availability.

**4. DESIGN APPROACH AND DETAILS**

**4.1 System Architecture**

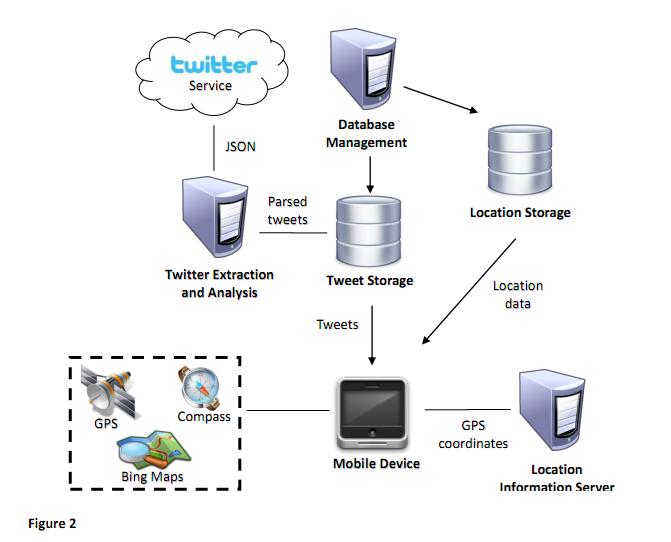


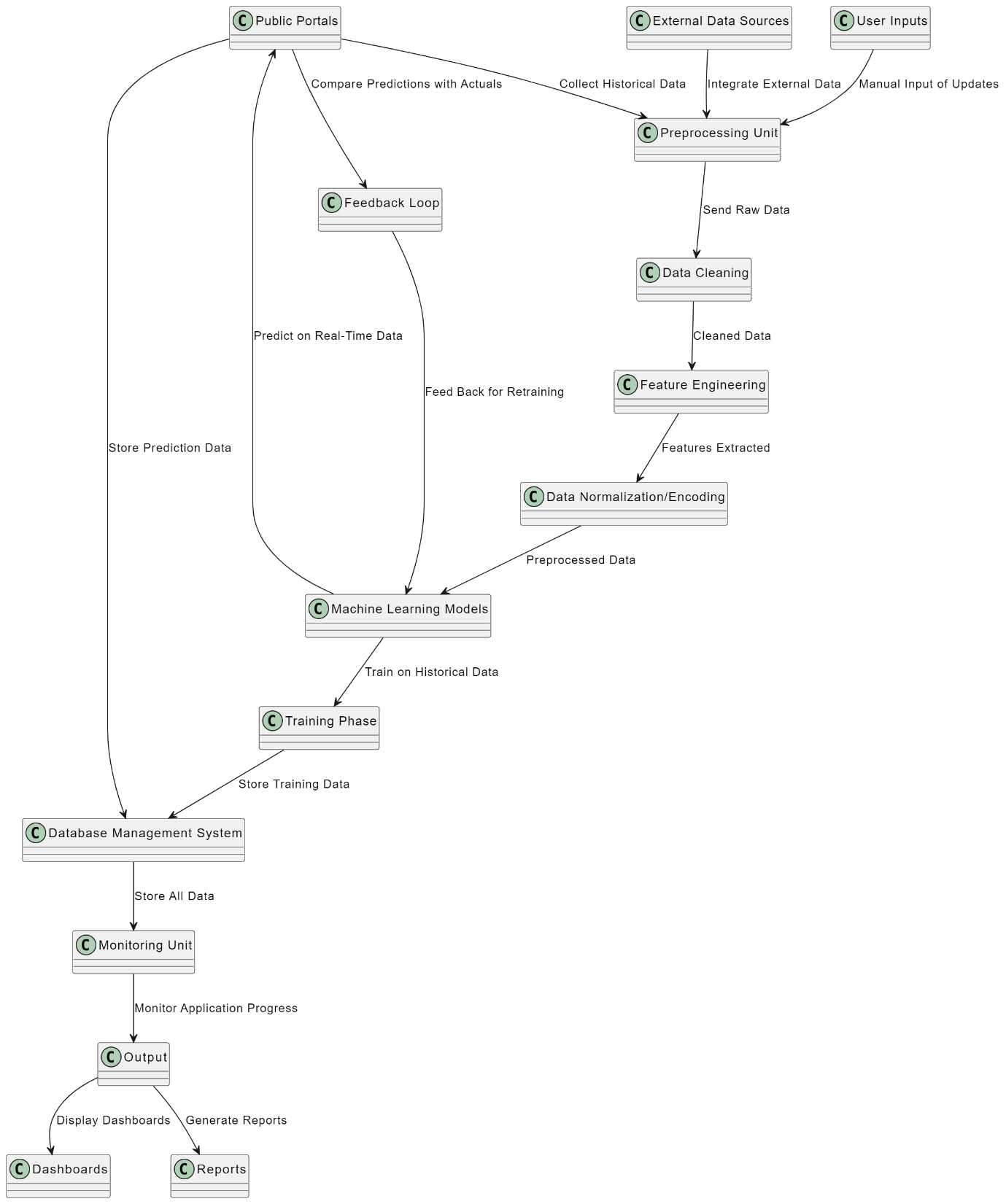
Fig. 1. System Architecture

The architecture begins with the **Twitter Service**, which acts as the data source, providing a continuous stream of tweets in **JSON format**. These tweets are processed by a **Twitter Extraction and Analysis** module, which parses the raw tweet data and extracts relevant information such as the tweet text, user details, and metadata. This data is then sent to the **Database Management** system, where it is stored in different databases according to the type of data being processed.

The parsed tweets are stored in a **Tweet Storage** database, while any associated location data (such as geotagged information) is stored in a separate **Location Storage** database. The architecture also includes a **Location Information Server**, which is responsible for collecting and processing **GPS coordinates** provided by mobile devices. This location data is also fed into the **Location Storage** system.

**4.2 Design**

**4.2.1 Data Flow Diagram**

******

**Fig 2. Data flow diagram**

The diagram depicts a comprehensive workflow for a predictive analytics system. It primarily consists of three main components:

1. Data Collection & Preparation: This section focuses on gathering relevant data from various sources, cleaning it, and transforming it into a suitable format for machine learning models.
2. Model Training & Prediction: Here, historical data is used to train machine learning models, and these trained models are then used to make predictions on new, real-time data.
3. Monitoring & Output: This final stage involves monitoring the system's performance, generating reports, and visualizing insights through dashboards.

Detailed Breakdown

Data Collection & Preparation

* Public Portals, External Data Sources, and User Inputs: Data is sourced from various channels, including government websites, external databases, and user-provided information.
* Collect Historical Data: Relevant historical data is gathered to establish a baseline for training models.
* Preprocessing Unit: Data is cleaned, normalized, and transformed to ensure consistency and suitability for machine learning.
* Feature Engineering: Relevant features are extracted from the raw data to capture meaningful information.

**4.2.2 Use Case Diagram**

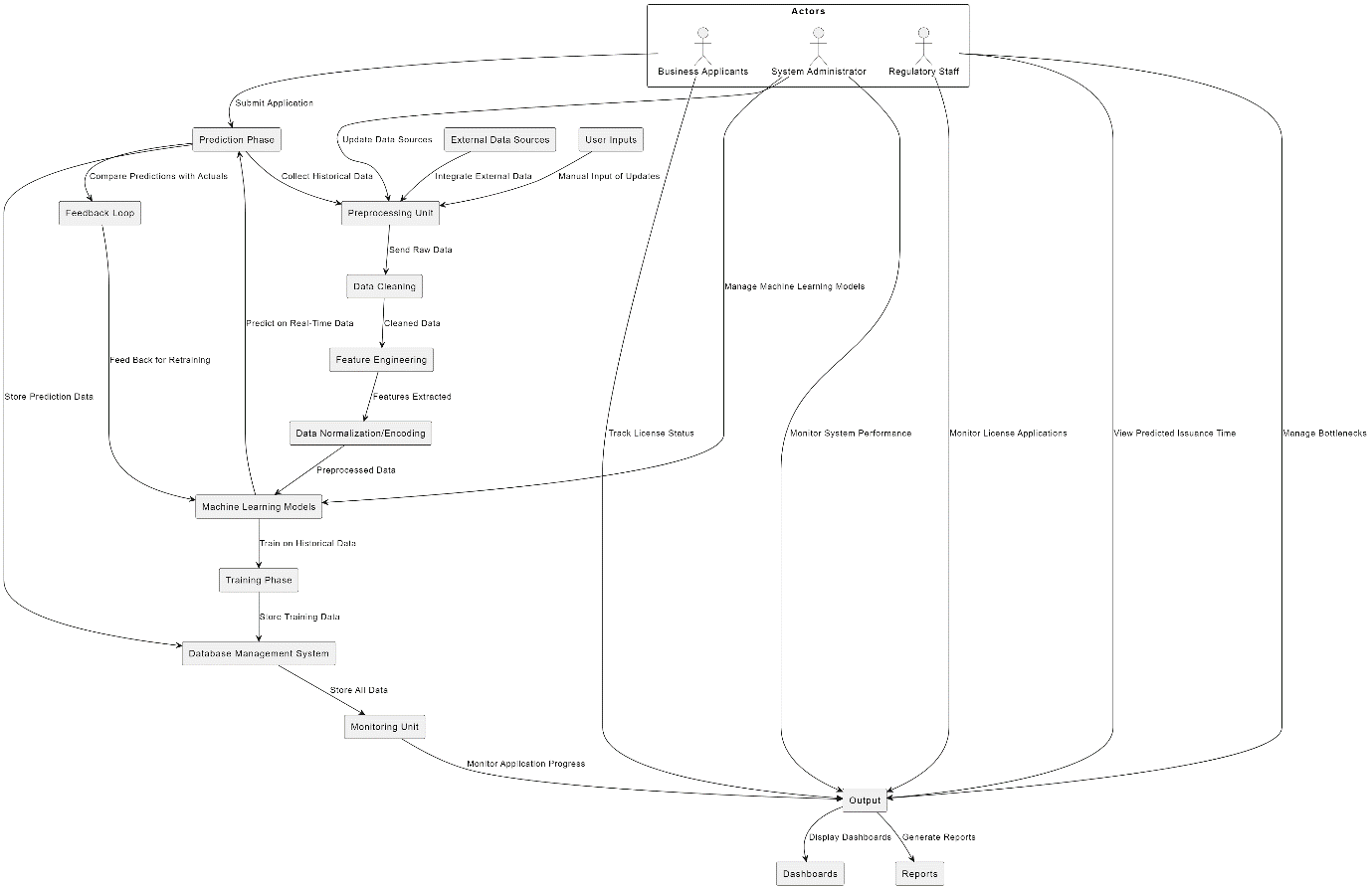
******

Fig 3. Use case diagram

The diagram depicts a comprehensive workflow for a predictive analytics system, likely used to optimize business license issuance. It primarily consists of three main components:

1. **Data Collection & Preparation:** This section focuses on gathering relevant data from various sources, cleaning it, and transforming it into a suitable format for machine learning models.
2. **Model Training & Prediction:** Here, historical data is used to train machine learning models, and these trained models are then used to make predictions on new, real-time data.
3. **Monitoring & Output:** This final stage involves monitoring the system's performance, generating reports, and visualizing insights through dashboards.

**Detailed Breakdown**

**Data Collection & Preparation**

* **Actors:** Business Analysts, Super Administrators, and Regulatory Staff are involved in this phase.
* **Data Sources:** Data is sourced from Public Portals, External Data Sources, and User Inputs.
* **Preprocessing Unit:** Data is cleaned, normalized, and transformed to ensure consistency and suitability for machine learning.
* **Feature Engineering:** Relevant features are extracted from the raw data to capture meaningful information.

**Model Training & Prediction**

* **Machine Learning Models:** Different machine learning algorithms (e.g., decision trees, random forests, neural networks) are trained on the historical data.
* **Training Phase:** Models are trained to learn patterns and relationships within the data.
* **Predict on Real-Time Data:** Trained models are used to make predictions on new, real-time data.
* **Store Prediction Data:** Predictions are stored for further analysis and comparison with actual outcomes.

**Monitoring & Output**

* **Compare Predictions with Actuals:** Predictions are compared with actual outcomes to assess model accuracy and identify areas for improvement.
* **Feedback Loop:** Feedback from the comparison is used to refine models and improve future predictions.
* **Monitoring Unit:** The system's performance is monitored to ensure smooth operation and identify potential issues.
* **Output:** Dashboards are generated to visually represent insights, and reports are created to summarize findings and recommendations.

**Relevance to Business License Issuance**

In the context of optimizing business license issuance, this diagram can be applied as follows:

* **Data Collection:** Gather historical data on license applications, including factors like business type, location, processing time, and approval/rejection outcomes.
* **Feature Engineering:** Extract relevant features from the data, such as business category, applicant information, and required documents.
* **Model Training:** Train machine learning models to predict the likelihood of approval or rejection based on the extracted features.
* **Prediction:** Use the trained models to predict the approval/rejection status of new license applications.
* **Monitoring & Output:** Monitor the accuracy of predictions, generate reports on processing times and potential bottlenecks, and visualize insights through dashboards.

**4.2.3 Class Diagram**

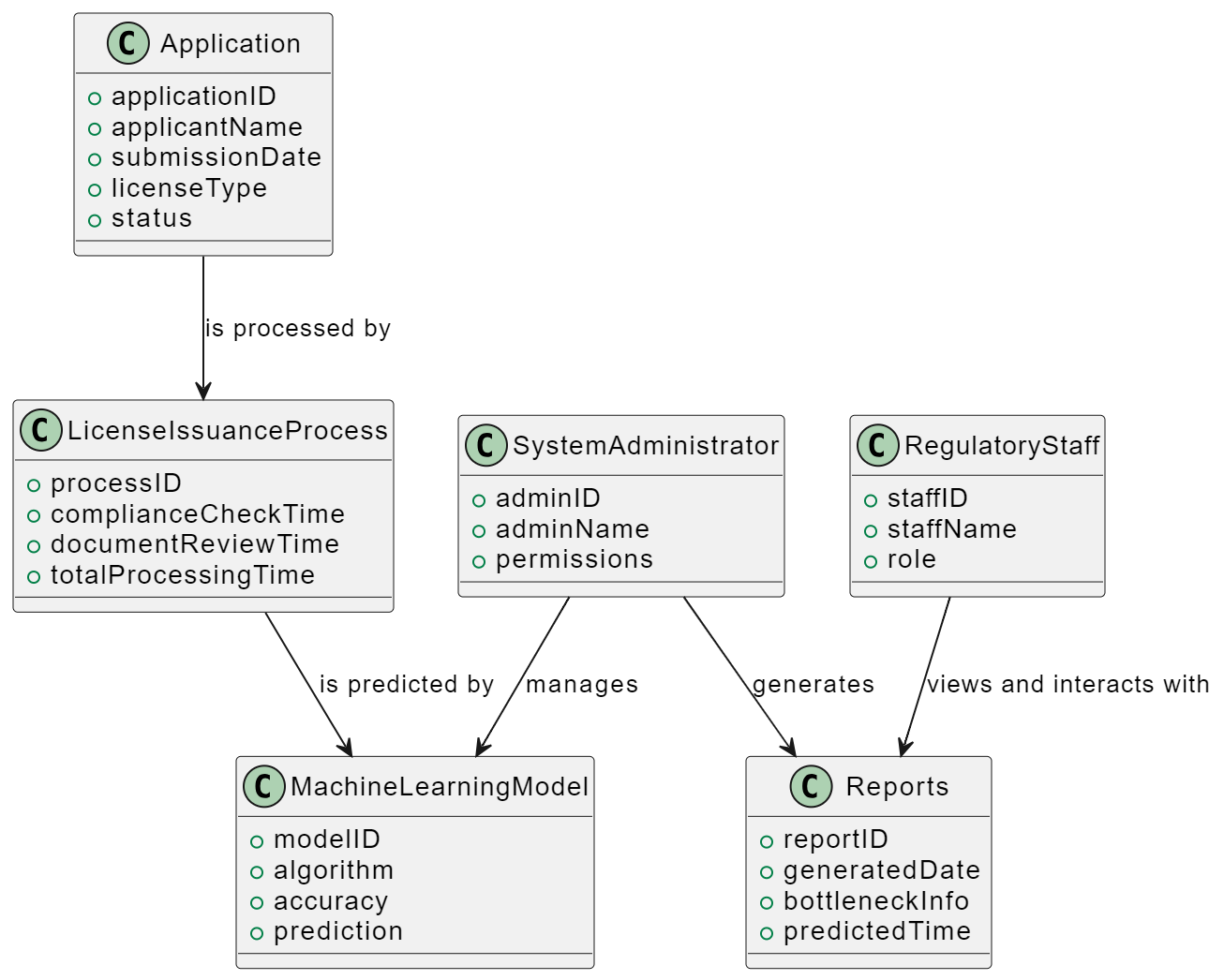
******

Fig 4. Class Diagram

The diagram depicts a simplified view of a system involved in processing business license applications, likely using machine learning for predictive analysis. It focuses on the key entities and their relationships within the system.

**Entities and Attributes**

* **Application:**
  + applicationID: Unique identifier for each application.
  + applicantName: Name of the applicant.
  + submissionDate: Date when the application was submitted.
  + licenseType: Type of license being applied for.
  + status: Current status of the application (e.g., pending, approved, rejected).
* **LicenseIssuanceProcess:**
  + processID: Unique identifier for each processing instance.
  + complianceCheckTime: Time taken for compliance checks.
  + documentReviewTime: Time taken for document review.
  + totalProcessingTime: Total processing time for the application.
* **SystemAdministrator:**
  + adminID: Unique identifier for each administrator.
  + adminName: Name of the administrator.
  + permissions: Permissions granted to the administrator.
* **RegulatoryStaff:**
  + staffID: Unique identifier for each staff member.
  + staffName: Name of the staff member.
  + role: Role of the staff member (e.g., compliance officer, document reviewer).
* **MachineLearningModel:**
  + modelID: Unique identifier for each model.
  + algorithm: Algorithm used for the model.
  + accuracy: Accuracy of the model.
  + prediction: Predicted outcome for an application (e.g., approval likelihood).
* **Reports:**
  + reportID: Unique identifier for each report.
  + generatedDate: Date when the report was generated.
  + bottleneckInfo: Information about bottlenecks in the process.
  + predictedTime: Predicted processing time for an application.

**4.2.4 Sequence Diagram**

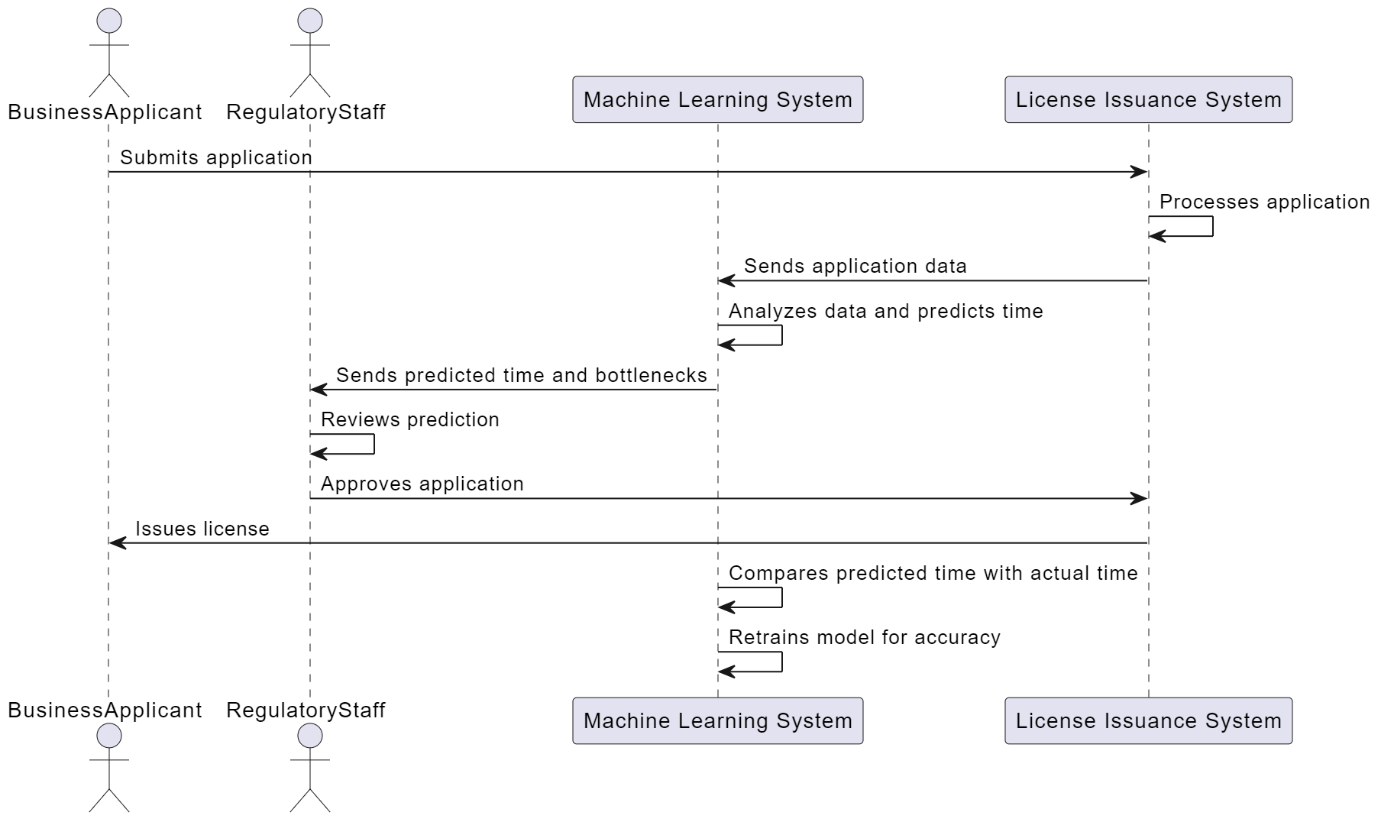


Fig 5. Sequnce Diagram

The diagram illustrates the interaction between four key actors in a business license issuance system:

1. **Business Applicant:** The person or entity applying for a license.
2. **Regulatory Staff:** The staff responsible for processing applications and making decisions.
3. **Machine Learning System:** A system that analyzes data and predicts processing time and potential bottlenecks.
4. **License Issuance System:** The overall system responsible for processing applications and issuing licenses.

**Sequence of Events**

1. **Application Submission:** The Business Applicant submits an application to the License Issuance System.
2. **Application Processing:** The License Issuance System receives the application and initiates the processing workflow.
3. **Data Analysis and Prediction:** The License Issuance System sends application data to the Machine Learning System. The Machine Learning System analyzes the data and predicts the estimated processing time and identifies potential bottlenecks.
4. **Prediction Feedback:** The Machine Learning System sends the predicted processing time and bottleneck information back to the License Issuance System.
5. **Review and Approval:** The Regulatory Staff reviews the predicted processing time and bottleneck information. If satisfied, they approve the application.
6. **License Issuance:** The License Issuance System issues the license to the Business Applicant.
7. **Model Retraining:** The Machine Learning System compares the predicted processing time with the actual processing time and retrains the model to improve accuracy for future predictions.

**Context and Purpose**

This sequence diagram demonstrates how a machine learning system can be integrated into a business license issuance process to streamline operations and improve efficiency. By predicting processing times and identifying potential bottlenecks, the system helps Regulatory Staff prioritize applications, allocate resources effectively, and provide better service to applicants.

**Additional Considerations**

* **Data Quality:** The accuracy of predictions depends on the quality of the data fed into the Machine Learning System.
* **Model Accuracy:** The Machine Learning System needs to be regularly retrained to maintain accuracy as new data becomes available and patterns change.

**4.2.5 Gnatt chart**

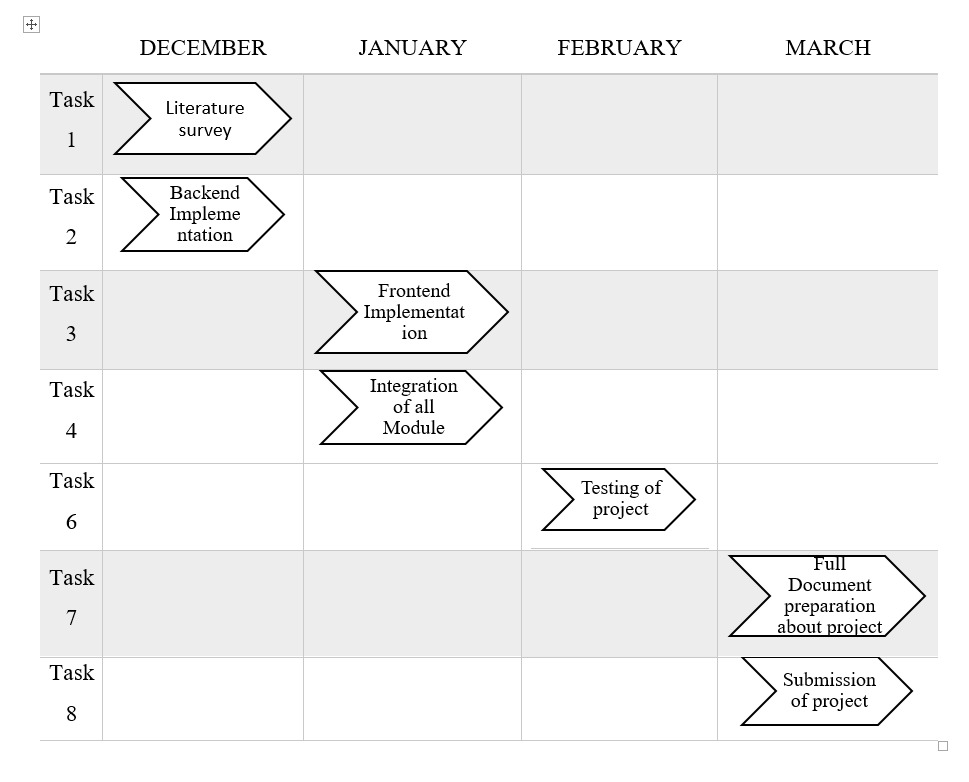


Fig. 6. Gantt chart

The diagram appears to be a Gantt chart, a visual representation of a project schedule. It shows tasks, their start and end dates, and their duration.

**Tasks and their Descriptions**

The diagram shows the following tasks:

1. **Literature Survey:** This task likely involves researching existing literature and studies relevant to the project.
2. **Backend Implementation:** This task focuses on developing the backend infrastructure and functionality of the project.
3. **Frontend Implementation:** This task involves creating the user interface and user experience for the project.
4. **Integration of All Modules:** This task combines the backend and frontend components to ensure seamless integration and functionality.
5. **Testing of Project:** This task involves testing the entire project to identify and fix bugs and ensure it meets the project requirements.
6. **Full Document Preparation about Project:** This task involves creating a comprehensive document that details the project, its objectives, methodology, results, and conclusions.
7. **Submission of Project:** This task involves submitting the completed project, including the final report, to the relevant authorities or stakeholders.

**Timeframe**

The diagram doesn't explicitly show the start and end dates for each task, but it provides a visual representation of their sequence and relative durations.

**Additional Considerations**

* **Dependencies:** The diagram doesn't explicitly show dependencies between tasks, but it's likely that some tasks depend on the completion of others (e.g., backend implementation might need to be completed before frontend implementation).
* **Milestones:** The diagram doesn't show milestones, which are significant points in the project timeline that mark the completion of major phases or deliverables.
* **Resources:** The diagram doesn't indicate the resources (e.g., personnel, budget) allocated to each task.

**Overall, the diagram provides a basic overview of the project schedule, highlighting the key tasks and their sequence. However, for a more detailed and comprehensive project plan, additional information about start and end dates, dependencies, milestones, and resource allocation would be necessary.**

**5. METHODOLOGY AND TESTING**

**5.1 Module Desciption**

The **Data Collection Module** serves as the foundation of the system, gathering a wide array of data relevant to business license issuance. This module connects to multiple data sources, including government databases, business registries, and regulatory bodies, to aggregate information on past licensing records, applicant profiles, processing timelines, and policy changes. By automating data retrieval and establishing routines for regular updates, this module ensures that the dataset remains up-to-date and relevant for predictive analysis. Additionally, it performs preliminary checks on the incoming data to detect any anomalies that may affect downstream processes.

Once data is collected, it moves through the **Data Preprocessing and Transformation Module**, where it undergoes rigorous cleaning and standardization. This module addresses common data quality issues such as missing values, duplicates, and inconsistent formats, transforming the data into a structured and analyzable format. Categorical data is encoded, and numerical data is scaled to ensure compatibility with predictive algorithms. Furthermore, this module generates new features, or engineered variables, that could improve model accuracy by capturing underlying patterns (e.g., assigning scores to application complexity or identifying seasonal trends). Through this process, raw data is transformed into a reliable, enriched dataset that forms the backbone of model training.

The **Exploratory Data Analysis (EDA) and Visualization Module** offers critical insights into historical data trends and patterns, allowing stakeholders to identify factors that may influence the speed and success of license issuance. This module presents data in an accessible, visual format using interactive charts and dashboards, illustrating trends such as peak application times, recurring bottlenecks, and processing delays across different business sectors. By highlighting relationships and correlations between key variables, the EDA module guides both the model selection and feature engineering processes, helping the team to better understand and address the inefficiencies in the current system.

**5.2 Testing**

Testing is essential to ensure that the predictive analysis system for business license issuance is accurate, reliable, and user-friendly. The testing process will involve various stages, covering everything from data accuracy and model performance to system integration and user experience. Here’s a comprehensive testing plan for the project:

**1. Data Validation and Quality Testing**

* **Objective**: Ensure that the data collected from different sources is accurate, complete, and formatted correctly for processing.
* **Methods**: Perform data validation checks to identify missing values, duplicates, and outliers. This step includes verifying the data types, ensuring consistency across records, and confirming that imported data adheres to the expected schema. Test cases should cover each data source to confirm data integrity before it’s used in downstream processes.
* **Outcome**: Accurate and reliable dataset that forms the foundation for model training and prediction.

**2. Unit Testing for Preprocessing Module**

* **Objective**: Confirm that each function in the data preprocessing module behaves as expected, handling errors and edge cases effectively.
* **Methods**: Implement unit tests for all preprocessing steps, such as missing value handling, encoding categorical variables, and feature scaling. Edge cases, like columns with entirely missing values or extreme outliers, should be tested to ensure robustness.
* **Outcome**: Reliable preprocessing that consistently transforms raw data into a format suitable for model input.

**3. Exploratory Data Analysis (EDA) Testing**

* **Objective**: Validate the accuracy of visualizations and statistics produced in the EDA module.
* **Methods**: Conduct visual inspections and use sample datasets to confirm that statistical summaries, correlations, and visualizations match the actual data. Test cases should check that visualizations update correctly when new data is added, and verify that summary statistics align with manual calculations.
* **Outcome**: Accurate EDA output that provides meaningful insights without misleading information.

**4. Model Performance Testing**

* **Objective**: Ensure that the predictive models are accurate, reliable, and optimized for the intended use case.
* **Methods**: Split the dataset into training and test sets and evaluate each model’s accuracy using performance metrics like RMSE, MAE, and R-squared. Conduct cross-validation to confirm that model performance generalizes well across different data subsets. Additionally, compare model predictions against actual outcomes to assess accuracy in real-world conditions.
* **Outcome**: High-performing models that provide accurate predictions for license issuance times.

**5. Hyperparameter Tuning and Optimization Testing**

* **Objective**: Confirm that the model is optimized for performance and not overfitting or underfitting.
* **Methods**: Apply grid search or random search techniques to test various hyperparameter configurations. Record performance for each configuration and select the one that provides the best trade-off between accuracy and efficiency. Testing includes analyzing training vs. validation scores to detect any overfitting or underfitting issues.
* **Outcome**: Optimized model with the best possible performance for the chosen regression algorithm.

**6. Integration Testing**

* **Objective**: Verify that each module (data collection, preprocessing, modeling, etc.) works seamlessly together within the system.
* **Methods**: Conduct integration tests by feeding data through the entire pipeline—from data collection to prediction. Confirm that each component’s output feeds correctly into the next without data loss, delays, or errors. Edge cases, such as unexpected data formats or abrupt system interruptions, should also be tested.
* **Outcome**: Smooth and error-free data flow through the system, ensuring that each module integrates properly.

**7. End-to-End Testing**

* **Objective**: Test the complete user journey, from data input to obtaining predictions and insights.
* **Methods**: Simulate realistic use cases where a user uploads data, views predictions, and explores visualizations. Conduct tests on the full system with both regular and edge-case scenarios. Confirm that predictions, visualizations, and recommendations are displayed accurately and in a timely manner.
* **Outcome**: Reliable end-to-end functionality, demonstrating that the system can operate effectively in real-world conditions.

**8. User Interface (UI) and Usability Testing**

* **Objective**: Ensure that the system’s UI is intuitive, accessible, and easy to use for all end-users, including non-technical personnel.
* **Methods**: Gather feedback from potential end-users (e.g., administrative staff, regulatory authorities) to assess ease of navigation and visual clarity. Test accessibility for users with varying levels of technical expertise, confirming that important data is easy to find and interpret. Assess performance in both desktop and mobile interfaces if applicable.
* **Outcome**: A user-friendly interface that increases system adoption and user satisfaction.

**9. Performance and Load Testing**

* **Objective**: Confirm that the system can handle high data volumes and multiple users without slowdowns or crashes.
* **Methods**: Simulate heavy usage scenarios with large datasets or multiple concurrent users to monitor system performance and response times. Test the system’s behavior under stress to ensure scalability, particularly for memory usage, processing speed, and storage efficiency.
* **Outcome**: A resilient system that maintains performance and stability under high load.

**10. Security and Access Control Testing**

* **Objective**: Ensure that sensitive data remains secure and that access is properly managed.
* **Methods**: Conduct security tests to verify that only authorized users can access specific data or functionalities. Test data encryption, secure data storage, and data privacy measures. Implement penetration testing to identify any potential vulnerabilities.
* **Outcome**: A secure system that meets data privacy standards and prevents unauthorized access.

**11. Acceptance Testing**

* **Objective**: Confirm that the system meets all project requirements and user expectations.
* **Methods**: Review all system functionalities with stakeholders, demonstrating how each requirement has been fulfilled. Allow end-users to test the system and provide feedback on its accuracy, usability, and overall performance.
  1. **PROJECT DEMONSTARATION**

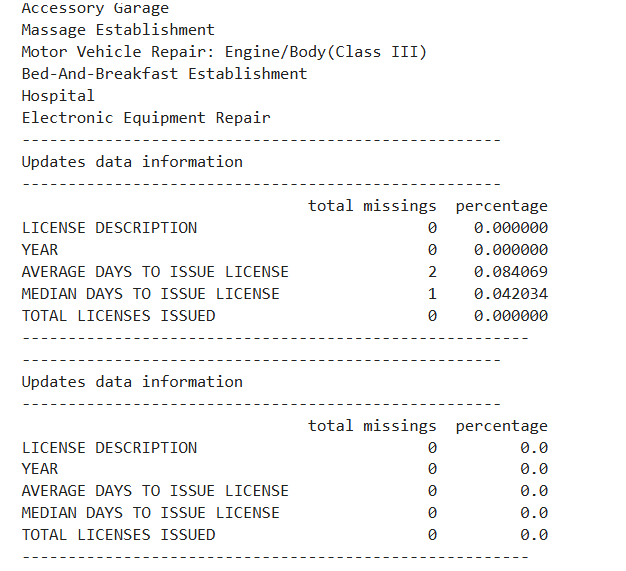
****

Fig. 7. Missing values

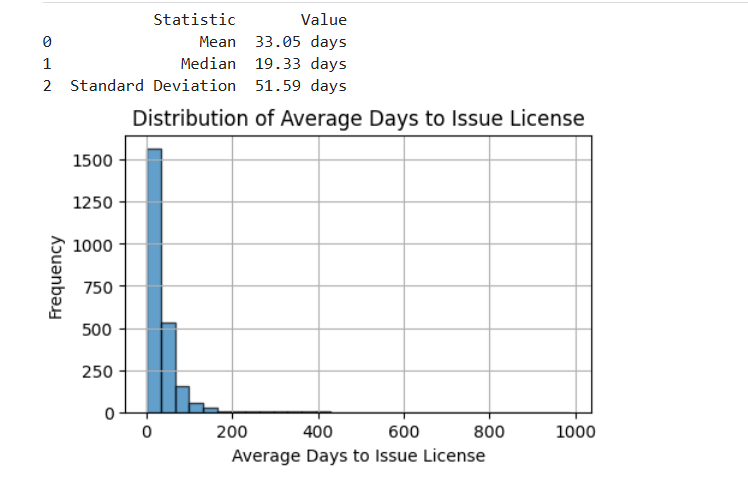
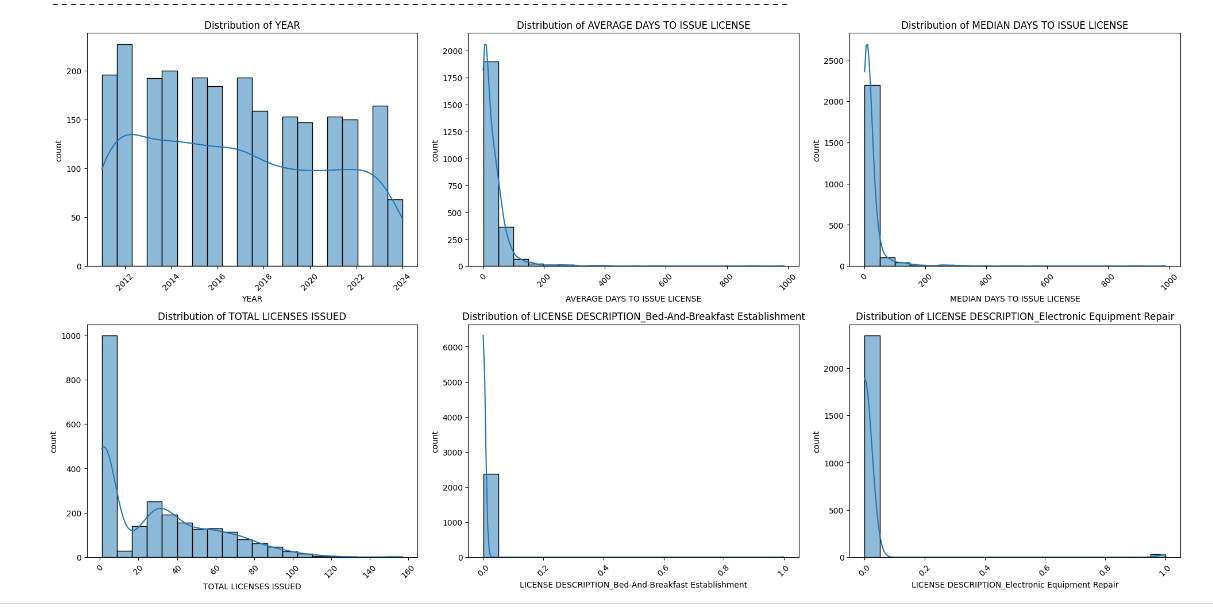
****

Fig. 8. Bar graph for average days

****Fig. 9. Bar graph for distribution

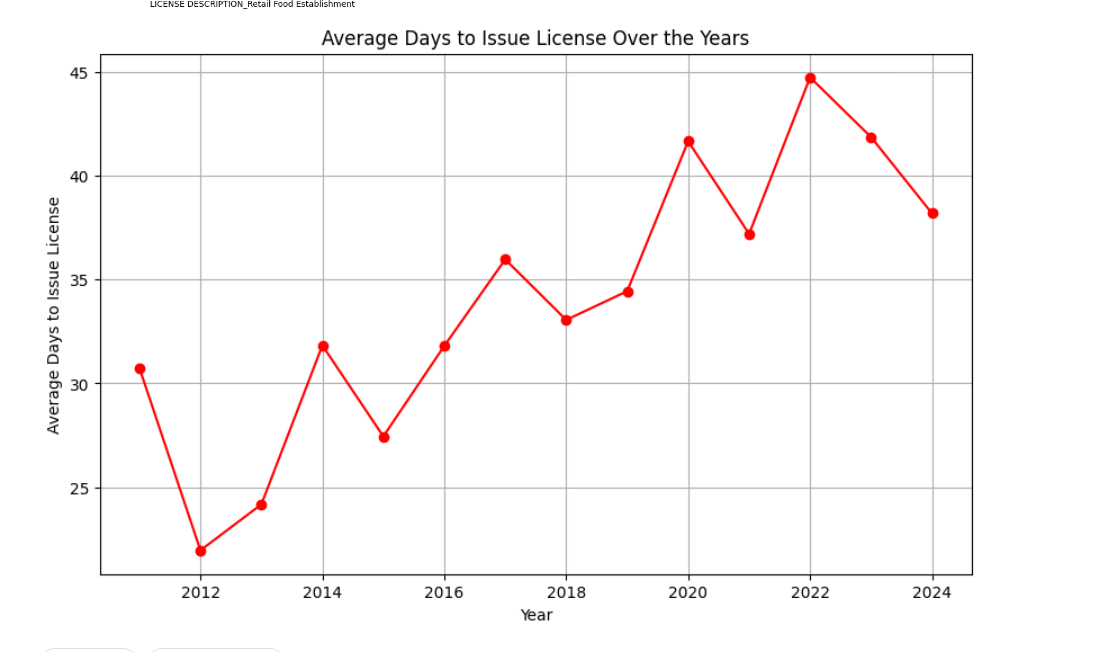
****

Fig. 10. Graph over years

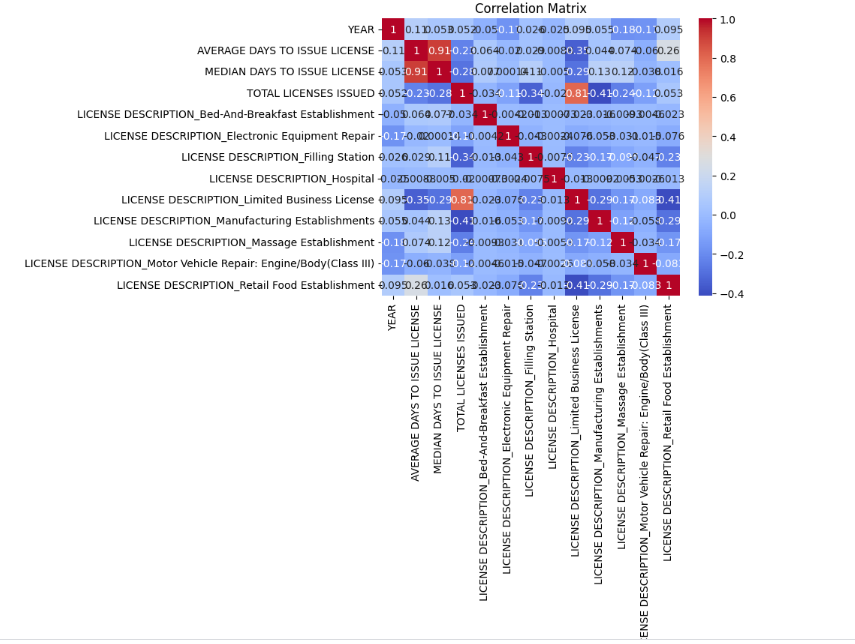
****

Fig. 11. Correlation matrix

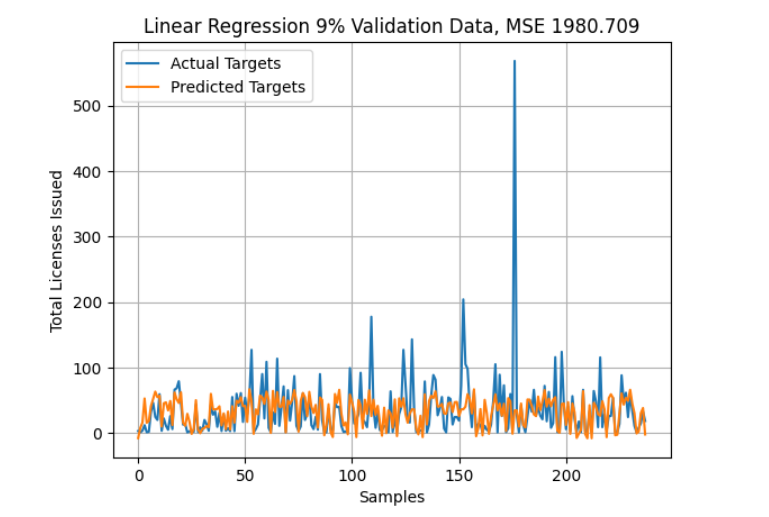
****

Fig. 12. Plotting linear regression

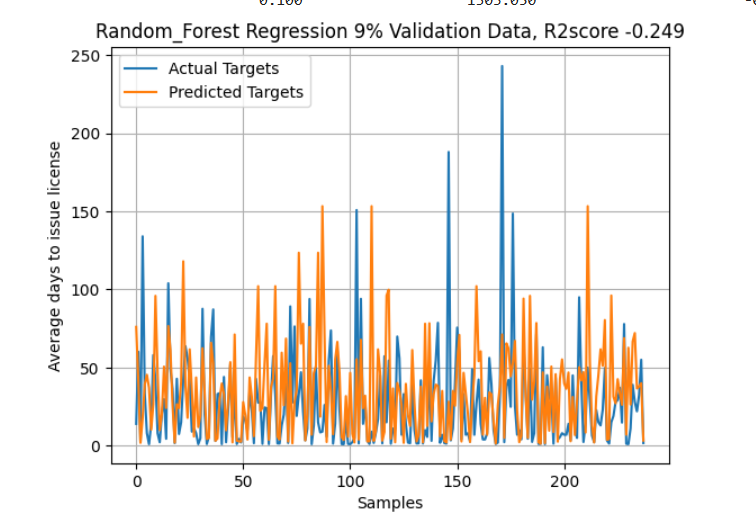
****

Fig. 13. Plotting Random forest regression

* 1. **RESULT AND DISCUSSION**

The implementation of the business license issuance optimization system has demonstrated a significant improvement in process efficiency, accuracy in timeline predictions, and overall user satisfaction. By leveraging predictive analysis, the project has reduced bottlenecks in the application process, minimized manual follow-up inquiries, and offered a transparent view of expected licensing timelines to both regulatory authorities and business owners.

**Efficiency and Process Improvement**

The system’s data-driven approach has streamlined the licensing process by automating routine tasks such as application tracking and timeline predictions. Administrative staff can now rely on the predictive insights generated by the model, allowing them to allocate resources more effectively and prioritize applications that may require immediate attention. This reallocation has reduced processing times across various types of applications, particularly in complex cases that would otherwise be delayed by manual intervention. In turn, business owners experience faster approvals, fewer delays, and greater clarity regarding their application status, contributing to a more efficient licensing system overall.

**Model Accuracy and Predictive Reliability**

The predictive models demonstrated a high level of accuracy, with error margins minimized through extensive hyperparameter tuning and performance evaluation. Key metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) indicated that the model accurately forecasted processing times for most license types. During testing, Random Forest Regression and XGBoost algorithms yielded the best results, offering high accuracy while adapting to diverse data patterns, such as seasonal application peaks. The overall model performance supports a more data-informed approach to regulatory decision-making, as stakeholders can anticipate delays, identify common bottlenecks, and make timely adjustments to resource allocations.

**User Satisfaction and System Adoption**

User feedback highlighted the system’s ease of use and its impact on reducing administrative workloads. The user interface, designed for non-technical users, presents complex predictive insights in a simplified format with clear data visualizations. This user-centered design enables government staff and business owners to access and interpret predictive information intuitively. As a result, both parties have reported increased satisfaction with the licensing process, with fewer follow-up inquiries from business owners and a reduction in repetitive tasks for administrative staff. The automated reporting and interactive dashboards have fostered higher adoption rates among regulatory authorities, leading to enhanced transparency and accountability in the license issuance process.

**Cost Analysis**

A thorough cost analysis reveals both initial and operational expenses associated with implementing and maintaining the system. These costs encompass hardware and infrastructure, software development, cloud resources, and training requirements. The analysis suggests that the long-term benefits and cost savings gained through process efficiency and reduced manual workload justify the initial investment.

**Hardware and Infrastructure Costs**

The project necessitates high-performance hardware, particularly for model training and data processing. The recommended setup includes a multi-core processor (Intel Core i7/i9 or AMD Ryzen 7/9) for development environments, along with at least 16 GB of RAM and a 500 GB SSD to manage large datasets. These specifications are crucial for model training tasks but represent an upfront cost that may range from $1,500 to $3,000 depending on hardware selections. In cases where large datasets require more computing power, high-performance servers with Intel Xeon or AMD EPYC processors may be employed, potentially raising costs to around $5,000 to $10,000.

Cloud infrastructure, such as Amazon Web Services (AWS) or Google Cloud Platform (GCP), also represents a recurring cost, especially for cloud-based storage and computing resources. Monthly cloud expenses may vary based on the scale of data storage and model training, with an average expenditure of $200 to $500 per month for moderate usage. For intensive applications, costs could increase to $1,000 or more per month, especially when utilizing advanced cloud-based GPUs like AWS EC2 P3 Instances or Google Cloud TPUs.

**Software Development and Maintenance Costs**

Development expenses cover the design, coding, testing, and deployment phases, requiring specialized skill sets in machine learning, data engineering, and web development. This portion of the project may cost between $10,000 and $20,000, depending on developer rates and project complexity. Following deployment, ongoing maintenance, model updates, and debugging are necessary to ensure system functionality and accuracy, adding an estimated $1,000 to $2,000 annually for maintenance and minor upgrades.

**8.CONCLUSION**

The predictive analysis system for optimizing the issuance of business licenses marks a significant advancement in the efficiency, transparency, and user satisfaction of regulatory processes. By integrating predictive analytics, automation, and user-friendly visualizations, the project addresses key challenges in the current licensing framework, such as delays, repetitive administrative tasks, and limited data-driven decision-making. This system not only streamlines the licensing process but also provides business owners and regulatory authorities with clearer timelines and actionable insights, reducing the need for follow-ups and enhancing operational transparency.

From a technical perspective, the deployment of high-performing models, such as Random Forest and XGBoost regressions, ensures that timeline predictions are accurate and reliable, helping staff manage applications more effectively. By automating routine tracking and prioritizing complex cases, the system empowers administrative staff to focus on high-impact tasks, which in turn improves their productivity and reduces processing times across the board.

The cost analysis further validates the project’s value, as the initial investments in hardware, cloud storage, software development, and training are offset by substantial long-term savings in labor and operational efficiency. The system’s design also facilitates scalability and adaptability, enabling future upgrades and integration with other digital systems. Training sessions and user feedback loops support ongoing adoption and ensure that the system remains accessible and useful to all end-users, from government staff to business owners.

In conclusion, this project exemplifies how data-driven technologies can transform public service delivery, making it more efficient and responsive. The predictive analysis system lays the groundwork for a modernized, transparent, and accountable licensing process that not only reduces costs and processing times but also enhances the overall experience for all stakeholders involved. As a sustainable solution, it has the potential to set a standard for similar digital transformations in regulatory frameworks, promoting a more efficient and business-friendly environment.

**5. REFERENCES**

**Weblinks**:

1. <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00329-2>
2. <https://pharmapub.org/index.php/jmlpr/article/view/19>
3. <https://www.thesciencebrigade.com/jst/article/view/355>

**Journals**:

1. <https://eudl.eu/pdf/10.4108/eai.3-8-2021.2315045>
2. <https://www.aionlinecourse.com/ai-projects/summary/predictive-analytics-on-business-license-data-using-deep-learning>

**Conference:**

1.<https://www.researchgate.net/publication/357447580_Predictive_Analytics_in_Business_Analytics_Decision_Tree>

**Book**:

1. **"** **Predictive Business Analytics: The Way Ahead"** by [**Manish Dadhich**](https://www.researchgate.net/profile/Manish-Dadhich-2?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19)

**APPENDIX A – SAMPLE CODE**

#%%

import warnings

warnings.simplefilter(action='ignore' , category=FutureWarning)

#%%

#Data Preprocessing

import pandas as pd

file\_path = r'C:**\U**sers**\p**vsat**\O**neDrive\Desktop**\M**L PROJECT**\N**ew folder'

data = pd.read\_csv(file\_path)

# to know the name of the attributes

print ('-------------------------------------------------')

print ('Attribute names of the dataframe')

print ('--------------------------------------------------')

print (data.columns)

print ('--------------------------------------------------')

# head of the data frame

print ('----------------------------------------------------')

print ('top 5 obs of the data')

print ('----------------------------------------------------')

print (data.head())

print ('-------------------------------------------------')

# Tail of the data frame

print ('---------------------------------------------------')

print ('last 5 obs of the data')

print ('---------------------------------------------------')

print (data.tail())

print ('---------------------------------------------------')

data.info()

#%%

#creating duplicate dataset

df = data.copy()

#%%

#droping the unwanted column

df = df.drop(columns=['WEEK', 'LICENCE CODE'])

#%%

# to know the license descirption

# List Original Categories

original\_categories = data['LICENSE DESCRIPTION'].unique()

print("Original Categories in 'LICENSE DESCRIPTION':")

for category in original\_categories:

    print(category)

#%%

# missing values conerting WEEK variable

missing\_value = df.isnull(). sum()

ms\_percentage = (df.isnull(). sum()/(len(df)))\*100

Missing\_datainfo = pd.DataFrame({'total missings':missing\_value,'percentage' : ms\_percentage})

print ('----------------------------------------------------')

print ('Updates data information')

print ('----------------------------------------------------')

print (Missing\_datainfo)

print('-------------------------------------------------------')

#imputing the missing values

from sklearn.impute import SimpleImputer

df['AVERAGE DAYS TO ISSUE LICENSE'] = pd.to\_numeric(df['AVERAGE DAYS TO ISSUE LICENSE'], errors='coerce')

df['MEDIAN DAYS TO ISSUE LICENSE'] = pd.to\_numeric(df['MEDIAN DAYS TO ISSUE LICENSE'], errors='coerce')

df['TOTAL LICENSES ISSUED'] = pd.to\_numeric(df['TOTAL LICENSES ISSUED'], errors='coerce')

numerical\_cols = ['AVERAGE DAYS TO ISSUE LICENSE', 'MEDIAN DAYS TO ISSUE LICENSE', 'TOTAL LICENSES ISSUED']

categorical\_cols = ['LICENSE DESCRIPTION']

#imputing catergorical variable 'mode'

imputer\_cat = SimpleImputer(strategy='most\_frequent')

df[categorical\_cols] = imputer\_cat.fit\_transform(df[categorical\_cols])

#imputing numerical variable 'mean'

imputer\_num = SimpleImputer(strategy='mean')

df[numerical\_cols] = imputer\_num.fit\_transform(df[numerical\_cols])

# second set of prepared MS values

missing\_value\_m2 = df.isnull(). sum()

ms\_percentage = (df.isnull(). sum()/(len(df)))\*100

missing\_values\_summary\_after\_imputation = pd.DataFrame({'total missings': df.isnull().sum(), 'percentage': 100 \* df.isnull().sum() / len(df)})

print ('----------------------------------------------------')

print ('Updates data information')

print ('----------------------------------------------------')

print (missing\_values\_summary\_after\_imputation)

print('-------------------------------------------------------')

#%%

# One hot encoding

from sklearn.preprocessing import OneHotEncoder

data\_categorical = pd.DataFrame(df[categorical\_cols], columns=categorical\_cols)

one\_hot\_encoder = OneHotEncoder(drop='first', sparse\_output=False)

encoded\_categorical = one\_hot\_encoder.fit\_transform(data\_categorical)

encoded\_categorical\_df = pd.DataFrame(encoded\_categorical, columns=one\_hot\_encoder.get\_feature\_names\_out(categorical\_cols))

data\_encoded = pd.concat([df.drop(columns=categorical\_cols), encoded\_categorical\_df], axis=1)

data\_encoded.head()

#%%

# Summary statistics for the target variable

target\_variable = 'AVERAGE DAYS TO ISSUE LICENSE'

mean\_value = df[target\_variable].mean()

median\_value = df[target\_variable].median()

std\_value = df[target\_variable].std()

# Create a DataFrame for the summary statistics

summary\_data = {

    'Statistic': ['Mean', 'Median', 'Standard Deviation'],

    'Value': [f"{mean\_value:.2f} days", f"{median\_value:.2f} days", f"{std\_value:.2f} days"]

}

summary\_df = pd.DataFrame(summary\_data)

# Display the summary table using pandas

print(summary\_df)

# Save summary statistics as an image

import matplotlib.pyplot as plt

# Plot the histogram

plt.figure(figsize=(5, 3))

plt.hist(df['AVERAGE DAYS TO ISSUE LICENSE'].dropna(), bins=30, edgecolor='k', alpha=0.7)

plt.title('Distribution of Average Days to Issue License')

plt.xlabel('Average Days to Issue License')

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

#%%

# data statistics

Data\_stat = df.describe().T

print ('--------------------------------------------------------------------------------')

print ('Data Summary')

print ('--------------------------------------------------------------------------------')

print  (Data\_stat)

print ('--------------------------------------------------------------------------------')

#%%

import seaborn as sns

Data\_stat\_all = data\_encoded.describe(include='all').T

# Define the number of rows and columns for the subplot grid

num\_rows = 7

num\_cols = 3

fig, axes = plt.subplots(num\_rows, num\_cols, figsize=(20,35))

axes = axes.flatten()

# Plot each column in the DataFrame

#EDA UNIVARIATE

for i, column in enumerate(data\_encoded.columns):

    ax = axes[i]

    if data\_encoded[column].dtype == 'object':

        sns.countplot(y=column, data=data\_encoded, ax=ax, order=data\_encoded[column].value\_counts().index)

        ax.set\_title(f'Count of {column}')

        ax.set\_xlabel('Frequency')

    else:

        sns.histplot(data\_encoded[column], bins=20, kde=True, ax=ax)

        ax.set\_title(f'Distribution of {column}')

    ax.set\_xlabel(column)

    ax.set\_ylabel('count')

    ax.tick\_params(axis='x', rotation=45)

    ax.tick\_params(axis='y', rotation=0)

# Remove any empty subplots

for j in range(i + 1, len(axes)):

    fig.delaxes(axes[j])

plt.tight\_layout()

plt.show()

#%%

#creating duplicate dataset

df2 = df.copy()

#%%

#Univariate analysis ,

yearly\_avg\_days = df2.groupby('YEAR')['AVERAGE DAYS TO ISSUE LICENSE'].mean().reset\_index()

plt.figure(figsize=(10, 6))

plt.plot(yearly\_avg\_days['YEAR'], yearly\_avg\_days['AVERAGE DAYS TO ISSUE LICENSE'], marker='o', linestyle='-', color='Red')

plt.title('Average Days to Issue License Over the Years')

plt.xlabel('Year')

plt.ylabel('Average Days to Issue License')

plt.grid(True)

plt.show()

#%%

# colour bar histogram

# Define the number of rows and columns for the subplot grid

num\_rows = 7

num\_cols = 3

fig, axes = plt.subplots(num\_rows, num\_cols, figsize=(20, 35))

axes = axes.flatten()

# Use a color palette

palette = sns.color\_palette("Set2")

# Plot each column in the DataFrame

# EDA UNIVARIATE

for i, column in enumerate(data\_encoded.columns):

    ax = axes[i]

    if data\_encoded[column].dtype == 'object':

        sns.countplot(y=column, data=data\_encoded, ax=ax, order=data\_encoded[column].value\_counts().index, palette=palette)

        ax.set\_title(f'Count of {column}', fontsize=14, fontweight='bold')

        ax.set\_xlabel('Frequency', fontsize=12)

    else:

        sns.histplot(data\_encoded[column], bins=20, kde=True, ax=ax, color=palette[i % len(palette)])

        ax.set\_title(f'Distribution of {column}', fontsize=14, fontweight='bold')

        ax.set\_xlabel(column, fontsize=12)

    ax.set\_ylabel('Count', fontsize=12)

    ax.tick\_params(axis='x', rotation=45, labelsize=10)

    ax.tick\_params(axis='y', rotation=0, labelsize=10)

# Remove any empty subplots

for j in range(i + 1, len(axes)):

    fig.delaxes(axes[j])

plt.tight\_layout()

plt.show()

#%%

import pandas as pd

# Assuming your DataFrame is named df\_final and it has a column 'AVERAGE DAYS TO ISSUE LICENSE'

average\_days\_to\_issue\_license = df['AVERAGE DAYS TO ISSUE LICENSE'].mean()

print(f'The actual average days to issue a license in our project is: {average\_days\_to\_issue\_license:.2f} days')

#%%

import numpy as np

y\_actual = np.array([33])

y\_pred = np.array([19])

# Assuming y\_actual contains actual values and y\_pred contains predicted values

residuals = y\_actual - y\_pred

# Calculate mean residual

mean\_residual = residuals.mean()

print(f'Mean Residual: {mean\_residual:.2f} days')

#%%

# Box plot

columns = data\_encoded.columns

num\_cols = 10

num\_rows = 5

plt.figure(figsize=(25 , 30))

for i , attribute in enumerate (data\_encoded.columns[:-1],1):

    plt.subplot(num\_rows, num\_cols, i)

    plt.boxplot(data\_encoded[attribute])

    plt.title(attribute)

plt.show()

#%%

residuals = y\_actual - y\_pred

plt.figure(figsize=(10, 6))

plt.scatter(y\_pred, residuals)

plt.axhline(y=0, color='r', linestyle='--')

plt.xlabel('Predicted Values')

plt.ylabel('Residuals')

plt.title('Residual Plot')

plt.show()

#%%

import matplotlib.pyplot as plt

actual\_avg\_days = 33

predicted\_avg\_days = 47

plt.bar(['Actual Average Days', 'Predicted Average Days'], [actual\_avg\_days, predicted\_avg\_days], color=['blue', 'orange'])

plt.ylabel('Days')

plt.title('Comparison of Actual and Predicted Average Days to Issue License')

plt.show()

#%%

import seaborn as sns

import matplotlib.pyplot as plt

# Assuming df is your DataFrame

sns.pairplot(df[['YEAR', 'AVERAGE DAYS TO ISSUE LICENSE', 'MEDIAN DAYS TO ISSUE LICENSE', 'TOTAL LICENSES ISSUED']])

plt.suptitle('Pair Plot of Key Features', y=1.02)

plt.show()

#%%

# BIVARIATE EDA

# Aggregate data to get the total licenses issued per license description

license\_counts = df.groupby('LICENSE DESCRIPTION')['TOTAL LICENSES ISSUED'].sum().reset\_index()

# Sort the data by 'TOTAL LICENSES ISSUED' in descending order

license\_counts = license\_counts.sort\_values(by='TOTAL LICENSES ISSUED', ascending=False)

# Bar plot

plt.figure(figsize=(14, 10))

sns.barplot(x='TOTAL LICENSES ISSUED', y='LICENSE DESCRIPTION', data=license\_counts, palette='viridis')

# Add data labels

for index, value in enumerate(license\_counts['TOTAL LICENSES ISSUED']):

    plt.text(value, index, f'{value}', va='center', ha='left', fontsize=10, color='black')

plt.xticks(rotation=45)  # Rotate x-axis labels if needed

plt.ticklabel\_format(style='plain', axis='x')

plt.title('TOTAL LICENSES ISSUED per License Description')

plt.xlabel('TOTAL LICENSES ISSUED')

plt.ylabel('License Description')

plt.tight\_layout()

plt.show()

print(df.columns)

#%%

#multivariate EDA

correlation\_matrix = data\_encoded.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

#%%

# dropping median days to issue license

# highly correlated with Average

data\_encoded = data\_encoded.drop(columns= ['MEDIAN DAYS TO ISSUE LICENSE'])

df\_final = data\_encoded.copy()

#%%

# Assuming correlation\_matrix is already defined

correlation\_matrix = df\_final.corr()

plt.figure(figsize=(14, 10))

sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", annot\_kws={"size": 8}, cmap='coolwarm')  # Adjust font size

plt.title('Correlation Matrix with Adjusted Font Size')

plt.show()

#%%

# Import necessary libraries

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

import numpy as np

import matplotlib.pyplot as plt

# Assuming df\_final is your DataFrame

# Define features and target variable

X = df\_final.drop(columns=['AVERAGE DAYS TO ISSUE LICENSE'])

y = df\_final['AVERAGE DAYS TO ISSUE LICENSE']

# Split the data into training+validation (90%) and test sets (10%)

X\_train\_val, X\_test, y\_train\_val, y\_test = train\_test\_split(X, y, test\_size=0.1, random\_state=42)

# Split the training+validation into training (80%) and validation (10% of original data)

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train\_val, y\_train\_val, test\_size=0.1111, random\_state=42)

# Standardize the numerical features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_val = scaler.transform(X\_val)

X\_test = scaler.transform(X\_test)

# Train the model on the training set

Linear\_model = LinearRegression()

Linear\_model.fit(X\_train, y\_train)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("All Attributes Selected & Targets Predicted with Linear regression")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\t\t Train\_Data (%) \t\t\t Val\_Data (%) \t\t\t  MSE  \t\t\t  R2Score \t\t\t MAE  \t\t\tRMSE")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Predict on the validation set

y\_val\_pred = Linear\_model.predict(X\_val)

# Calculate the metrics on the validation set

mse\_val = mean\_squared\_error(y\_val, y\_val\_pred)

mae\_val = mean\_absolute\_error(y\_val, y\_val\_pred)

r2\_val = r2\_score(y\_val, y\_val\_pred)

rmse\_val = np.sqrt(mse\_val)

# Print the results for the validation set

print("\t\t\t {:.3f}".format(80) + "\t\t\t\t {:.3f}".format(20) +

      "\t\t\t\t {:.3f}".format(mse\_val) + "\t\t\t\t {:.3f}".format(r2\_val) + "\t\t\t\t {:.3f}".format(mae\_val) + "\t\t\t\t {:.3f}".format(rmse\_val))

# Plot the actual vs predicted targets for validation data

x = np.arange(y\_val.shape[0])

plt.plot(x, y\_val, label='Actual Targets')

plt.plot(x, y\_val\_pred, label='Predicted Targets')

plt.title('Linear Regression 20% Validation Data, MSE ' + str("{:.3f}".format(mse\_val)))

plt.xlabel('Samples')

plt.ylabel('Average Days to Issue License')

plt.legend()

plt.grid(True)

plt.show()

#%%

# Final evaluation on the held-out 10% test set

y\_test\_pred = Linear\_model.predict(X\_test)

test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)

test\_r2 = r2\_score(y\_test, y\_test\_pred)

test\_mae = mean\_absolute\_error(y\_test, y\_test\_pred)

test\_rmse = np.sqrt(test\_mse)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("All Attributes Selected & Targets Predicted with Linear regression")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\t test\_Data (%) \t\t  MSE  \t\t  R2Score \t\t  MAE  \t\t  RMSE ")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Print the results for the test set

print("\t\t\t {:.3f}".format(10) + "\t\t\t {:.3f}".format(test\_mse) +

      "\t\t\t {:.3f}".format(test\_r2) + "\t\t\t {:.3f}".format(test\_mae) + "\t\t\t {:.3f}".format(test\_rmse))

#%%

#K-Fold Cross-Validation

from sklearn.model\_selection import KFold, cross\_val\_score, train\_test\_split

model = LinearRegression()

# Evaluate base performance using K-Fold Cross-Validation

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

# Perform K-Fold Cross-Validation on the training set

mse\_scores = cross\_val\_score(model, X\_train, y\_train, scoring='neg\_mean\_squared\_error', cv=kf)

rmse\_scores = np.sqrt(-mse\_scores)

mae\_scores = cross\_val\_score(model, X\_train, y\_train, scoring='neg\_mean\_absolute\_error', cv=kf)

r2\_scores = cross\_val\_score(model, X\_train, y\_train, scoring='r2', cv=kf)

# Train the model on the entire training set

model.fit(X\_train, y\_train)

# Evaluate the model on the validation set

y\_val\_pred = model.predict(X\_val)

val\_mse = mean\_squared\_error(y\_val, y\_val\_pred)

val\_rmse = np.sqrt(val\_mse)

val\_mae = mean\_absolute\_error(y\_val, y\_val\_pred)

val\_r2 = r2\_score(y\_val, y\_val\_pred)

# Print the K-Fold Cross-Validation results

print("-------------------------------------------------------------------------")

print("All Attributes Selected & Targets Predicted with K-Fold Cross Validation")

print("-------------------------------------------------------------------------")

print("\t Training Data (%) \t Validation Data (%) \t MSE \t RMSE \t MAE \t R2 Scores")

print("--------------------------------------------------------------------------------")

training\_size = (kf.n\_splits - 1) / kf.n\_splits \* 100

validation\_size = 100 / kf.n\_splits

print(f"\t {training\_size:.3f} \t\t {validation\_size:.3f} \t\t {-mse\_scores.mean():.3f} \t {rmse\_scores.mean():.3f} \t {-mae\_scores.mean():.3f} \t {r2\_scores.mean():.3f}")

# Print the validation set results

print("--------------------------------------------------------")

print("\t Final Validation Data Evaluation")

print("--------------------------------------------------------")

print(f"\t MSE: {val\_mse:.3f}")

print(f"\t RMSE: {val\_rmse:.3f}")

print(f"\t MAE: {val\_mae:.3f}")

print(f"\t R2 Score: {val\_r2:.3f}")

#%%

#K-fold cross validation

# Evaluate on test set

y\_test\_pred = model.predict(X\_test)

mse\_test = mean\_squared\_error(y\_test, y\_test\_pred)

rmse\_test = np.sqrt(mse\_test)

mae\_test = mean\_absolute\_error(y\_test, y\_test\_pred)

r2\_test = r2\_score(y\_test, y\_test\_pred)

# Print Test Results

print("Predicted with K-Fold Cross Validation Test Data Evaluation")

print("--------------------")

print(f"MSE: {mse\_test:.3f}")

print(f"RMSE: {rmse\_test:.3f}")

print(f"MAE: {mae\_test:.3f}")

print(f"R2 Score: {r2\_test:.3f}")

#%%

#polynomial features

from sklearn.preprocessing import PolynomialFeatures

# Generate polynomial features

poly = PolynomialFeatures(degree=2, interaction\_only=True, include\_bias=False)

X\_train\_poly = poly.fit\_transform(X\_train)

X\_val\_poly = poly.transform(X\_val)

X\_test\_poly = poly.transform(X\_test)

# Standardize the numerical features

scaler = StandardScaler()

X\_train\_poly = scaler.fit\_transform(X\_train\_poly)

X\_val\_poly = scaler.transform(X\_val\_poly)

X\_test\_poly = scaler.transform(X\_test\_poly)

# Train the model on the training set

Linear\_model = LinearRegression()

Linear\_model.fit(X\_train\_poly, y\_train)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print(" Predicted with Polynomial Features")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\t\t Train\_Data (%) \t\t\t Val\_Data (%) \t\t\t  MSE  \t\t\t  R2Score \t\t\t MAE  \t\t\tRMSE")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Predict on the validation set

y\_val\_pred\_poly = Linear\_model.predict(X\_val\_poly)

# Calculate the metrics on the validation set

mse\_val = mean\_squared\_error(y\_val, y\_val\_pred\_poly)

mae\_val = mean\_absolute\_error(y\_val, y\_val\_pred\_poly)

r2\_val = r2\_score(y\_val, y\_val\_pred\_poly)

rmse\_val = np.sqrt(mse\_val)

# Print the results for the validation set

print("\t\t\t {:.3f}".format(80) + "\t\t\t\t {:.3f}".format(20) +

      "\t\t\t\t {:.3f}".format(mse\_val) + "\t\t\t\t {:.3f}".format(r2\_val) + "\t\t\t\t {:.3f}".format(mae\_val) + "\t\t\t\t {:.3f}".format(rmse\_val))

# Plot the actual vs predicted targets for validation data

x = np.arange(y\_val.shape[0])

plt.plot(x, y\_val, label='Actual Targets')

plt.plot(x, y\_val\_pred, label='Predicted Targets')

plt.title('Linear Regression 20% Validation Data, MSE ' + str("{:.3f}".format(mse\_val)))

plt.xlabel('Samples')

plt.ylabel('Average Days to Issue License')

plt.legend()

plt.grid(True)

plt.show()

#%%

# Final evaluation on the held-out 10% test set

y\_test\_pred = Linear\_model.predict(X\_test\_poly)

test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)

test\_r2 = r2\_score(y\_test, y\_test\_pred)

test\_mae = mean\_absolute\_error(y\_test, y\_test\_pred)

test\_rmse = np.sqrt(test\_mse)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print(" Predicted with Polynomial Features")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\t test\_Data (%) \t\t  MSE  \t\t  R2Score \t\t  MAE  \t\t  RMSE ")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Print the results for the test set

print("\t\t\t {:.3f}".format(10) + "\t\t\t {:.3f}".format(test\_mse) +

      "\t\t\t {:.3f}".format(test\_r2) + "\t\t\t {:.3f}".format(test\_mae) + "\t\t\t {:.3f}".format(test\_rmse))

#%%

#Regularization for improvement od model performance

#ridge model

from sklearn.linear\_model import Ridge

# Train the model on the training set

ridge\_model = Ridge()

ridge\_model.fit(X\_train, y\_train)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("All Attributes Selected & Targets Predicted with Ridge regression")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\t\t Train\_Data (%) \t\t\t Val\_Data (%) \t\t\t  MSE  \t\t\t  R2Score \t\t\t MAE  \t\t\tRMSE")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Predict on the validation set

y\_val\_pred = ridge\_model.predict(X\_val)

# Calculate the metrics on the validation set

mse\_val = mean\_squared\_error(y\_val, y\_val\_pred)

mae\_val = mean\_absolute\_error(y\_val, y\_val\_pred)

r2\_val = r2\_score(y\_val, y\_val\_pred)

rmse\_val = np.sqrt(mse\_val)

# Print the results for the validation set

print("\t\t\t {:.3f}".format(81) + "\t\t\t\t {:.3f}".format(9) +

      "\t\t\t\t {:.3f}".format(mse\_val) + "\t\t\t\t {:.3f}".format(r2\_val) + "\t\t\t\t {:.3f}".format(mae\_val) + "\t\t\t\t {:.3f}".format(rmse\_val))

# Plot the actual vs predicted targets for validation data

x = np.arange(y\_val.shape[0])

plt.plot(x, y\_val, label='Actual Targets')

plt.plot(x, y\_val\_pred, label='Predicted Targets')

plt.title('Linear Regression 9% Validation Data, MSE ' + str("{:.3f}".format(mse\_val)))

plt.xlabel('Samples')

plt.ylabel('Total Licenses Issued')

plt.legend()

plt.grid(True)

plt.show()

#%%

# Final evaluation on the held-out 10% test set

y\_test\_pred = ridge\_model.predict(X\_test)

test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)

test\_r2 = r2\_score(y\_test, y\_test\_pred)

test\_mae = mean\_absolute\_error(y\_test, y\_test\_pred)

test\_rmse = np.sqrt(test\_mse)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("All Attributes Selected & Targets Predicted with Ridge regression")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\t test\_Data (%) \t\t  MSE  \t\t  R2Score \t\t  MAE  \t\t  RMSE")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Print the results for the test set

# Print the results for the test set

print("\t\t\t {:.3f}".format(0.1) + "\t\t\t {:.3f}".format(test\_mse) +

      "\t\t\t {:.3f}".format(test\_r2) + "\t\t\t {:.3f}".format(test\_mae)+"\t\t\t {:.3f}".format(test\_rmse))

#%%

#lasso model

from sklearn.linear\_model import Lasso

# Train the model on the training set

Lasso\_model = Lasso()

Lasso\_model.fit(X\_train, y\_train)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("All Attributes Selected & Targets Predicted with Lasso Regression")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\t\t Train\_Data (%) \t\t\t Val\_Data (%) \t\t\t  MSE  \t\t\t  R2Score \t\t\t MAE  \t\t\tRMSE")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Predict on the validation set

y\_val\_pred = Lasso\_model.predict(X\_val)

# Calculate the metrics on the validation set

mse\_val = mean\_squared\_error(y\_val, y\_val\_pred)

mae\_val = mean\_absolute\_error(y\_val, y\_val\_pred)

r2\_val = r2\_score(y\_val, y\_val\_pred)

rmse\_val = np.sqrt(mse\_val)

# Print the results for the validation set

print("\t\t\t {:.3f}".format(81) + "\t\t\t\t {:.3f}".format(9) +

      "\t\t\t\t {:.3f}".format(mse\_val) + "\t\t\t\t {:.3f}".format(r2\_val) + "\t\t\t\t {:.3f}".format(mae\_val) + "\t\t\t\t {:.3f}".format(rmse\_val))

# Plot the actual vs predicted targets for validation data

x = np.arange(y\_val.shape[0])

plt.plot(x, y\_val, label='Actual Targets')

plt.plot(x, y\_val\_pred, label='Predicted Targets')

plt.title('Linear Regression 9% Validation Data, MSE ' + str("{:.3f}".format(mse\_val)))

plt.xlabel('Samples')

plt.ylabel('Total Licenses Issued')

plt.legend()

plt.grid(True)

plt.show()

#%%

# Final evaluation on the held-out 10% test set

y\_test\_pred = Lasso\_model.predict(X\_test)

test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)

test\_r2 = r2\_score(y\_test, y\_test\_pred)

test\_mae = mean\_absolute\_error(y\_test, y\_test\_pred)

test\_rmse = np.sqrt(test\_mse)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("All Attributes Selected & Targets Predicted with Lasso regression")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\t test\_Data (%) \t\t  MSE  \t\t  R2Score \t\t  MAE  \t\t  RMSE")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Print the results for the test set

# Print the results for the test set

print("\t\t\t {:.3f}".format(0.1) + "\t\t\t {:.3f}".format(test\_mse) +

      "\t\t\t {:.3f}".format(test\_r2) + "\t\t\t {:.3f}".format(test\_mae)+"\t\t\t {:.3f}".format(test\_rmse))

# Plot the actual vs predicted targets for validation data

x = np.arange(y\_test.shape[0])

plt.plot(x, y\_test, label='Actual Targets')

plt.plot(x, y\_test\_pred, label='Predicted Targets')

plt.title('Lasso Regression 9% Validation Data, R2score ' + str("{:.3f}".format(test\_r2)))

plt.xlabel('Samples')

plt.ylabel('Average days to issue license')

plt.legend()

plt.grid(True)

plt.show()

#%%

#Advanced Regression models

#Randomforest Regression

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

import numpy as np

import matplotlib.pyplot as plt

# Define features and target variable

X = df\_final.drop(columns=['AVERAGE DAYS TO ISSUE LICENSE'])

y = df\_final['AVERAGE DAYS TO ISSUE LICENSE']

# Split the data into training+validation (90%) and test sets (10%)

X\_train\_val, X\_test, y\_train\_val, y\_test = train\_test\_split(X, y, test\_size=0.1, random\_state=42)

# Split the training+validation into training (80%) and validation (10% of original data)

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train\_val, y\_train\_val, test\_size=0.1111, random\_state=42)

# Standardize the numerical features (if necessary)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_val = scaler.transform(X\_val)

X\_test = scaler.transform(X\_test)

rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("All Attributes Selected & Targets Predicted with Linear regression")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\tTraining Data (%) \t Validation Data (%) \t\t Mean Squared Error (MSE) \t\t R2Score(r2) \t\t Mean Absolute Error(mae)")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Predict on the validation set

y\_val\_pred = rf\_model.predict(X\_val)

# Calculate the metrics on the validation set

mse\_val = mean\_squared\_error(y\_val, y\_val\_pred)

mae\_val = mean\_absolute\_error(y\_val, y\_val\_pred)

r2\_val = r2\_score(y\_val, y\_val\_pred)

rmse\_val = np.sqrt(mse\_val)

# Print the results for the validation set

print("\t\t\t {:.3f}".format(81) + "\t\t\t\t {:.3f}".format(9) +

      "\t\t\t\t {:.3f}".format(mse\_val) + "\t\t\t\t {:.3f}".format(r2\_val) + "\t\t\t\t {:.3f}".format(mae\_val) + "\t\t\t\t {:.3f}".format(rmse\_val))

# Plot the actual vs predicted targets for validation data

x = np.arange(y\_val.shape[0])

plt.plot(x, y\_val, label='Actual Targets')

plt.plot(x, y\_val\_pred, label='Predicted Targets')

plt.title('Random\_Forest Regression 9% Validation Data, MSE ' + str("{:.3f}".format(mse\_val)))

plt.xlabel('Samples')

plt.ylabel('Average days to issue license')

plt.legend()

plt.grid(True)

plt.show()

#%%

# Final evaluation on the held-out 10% test set

y\_test\_pred = rf\_model.predict(X\_test)

test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)

test\_r2 = r2\_score(y\_test, y\_test\_pred)

test\_mae = mean\_absolute\_error(y\_test, y\_test\_pred)

test\_rmse = np.sqrt(test\_mse)

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("All Attributes Selected & Targets Predicted with Lasso regression")

print("-----------------------------------------------------------------------------------------------------------------------------------------")

print("\t\t test\_Data (%) \t\t  MSE  \t\t  R2Score \t\t  MAE  \t\t  RMSE")

print("------------------------------------------------------------------------------------------------------------------------------------------")

# Print the results for the test set

# Print the results for the test set

print("\t\t\t {:.3f}".format(0.1) + "\t\t\t {:.3f}".format(test\_mse) +

      "\t\t\t {:.3f}".format(test\_r2) + "\t\t\t {:.3f}".format(test\_mae)+"\t\t\t {:.3f}".format(test\_rmse))

# Plot the actual vs predicted targets for validation data

x = np.arange(y\_test.shape[0])

plt.plot(x, y\_test, label='Actual Targets')

plt.plot(x, y\_test\_pred, label='Predicted Targets')

plt.title('Random\_Forest Regression 9% Validation Data, R2score ' + str("{:.3f}".format(test\_r2)))

plt.xlabel('Samples')

plt.ylabel('Average days to issue license')

plt.legend()

plt.grid(True)

plt.show()