DIRECT MARKET ACCESS FOR FARMERS APPLICATION DEVELOPMENT USING MACHINE LEARNING

A PROJECT REPORT

Submitted by,

AKSHAY B - 20211ISR0057
KARTHIK M SWAMY - 20211ISR0060
JERA PRAKASH - 20211ISR0076

Under the guidance of,

Dr. SIVARAMAKRISHNAN S
Associate professor

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION SCIENCE AND ENGINEERING (AI AND ROBOTICS).

At

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report **"DIRECT MARKET ACCESS FOR FARMERS APPLICATION DEVELOPMENT USING MACHINE LEARNING"** being submitted by "AKSHAY B , JERA PRAKASH , KARTHIK M SWAMY " bearing roll number(s) " 20211ISR0057 , 20211ISR0076 , 20211ISR0060" in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a Bonafide work carried out under my supervision.

Dr. SIVARAMAKRISHNAN. S
Associate Professor, & Guide
School of CSE&IS
Presidency University

Dr . ZAFAR ALI KHAN
Professor & HoD
School of CSE&IS
Presidency University

Dr. MYDHILI NAIR
Associate Dean
School of CSE
Presidency University

Dr. MD. SAMEERUDDIN KHAN
Pro-Vc School of Engineering
Dean -School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **DIRECT MARKET ACCESS FOR FARMERS APPLICATION DEVELOPMENT USING MACHINE LEARNING** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr. Sivaramakrishnan . S , Associate Professor, School of Computer Science Engineering Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Name		Roll No	Signatures
AKSHAY B	-	20211ISR0057	
KARTHIK M SWAMY	-	20211ISR0060	
JERA PRAKASH	-	20211ISR0076	

ABSTRACT

This paper introduces a crop recommendation system based on machine learning that uses soil and environmental characteristics to suggest which crop would be best to produce. Using a labelled agriculture dataset, we investigated and trained Random Forest and Support Vector Machine (SVM) classifiers. Random Forest was chosen as the foundation model since it demonstrated the highest accuracy among them. trained a TensorFlow neural network to replicate the Random Forest's predictions in order to facilitate mobile deployment. The TensorFlow Lite (.tflite) format was then created from this neural model. For Android apps, the TFLite model guarantees quick, offline crop prediction. By providing farmers with trustworthy crop recommendations, this method enhances decision-making and productivity. Additionally, it uses mobile technologies to bridge the gap between AI and rural farming techniques

KEYWORDS: Random Forest Classifier ml model, SVM ml model, TensorFlow Lite, Machine Learning, Android studio.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time. We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair,** School of Computer Science Engineering & Information Science, Presidency University, and **Dr. ZAFER ALI KHAN,** Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully. We are greatly indebted to our guide **Dr. Sivaramakrishnan. S, Associate professor ,** School of Computer Science Engineering & Information Science, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the PIP4004 Capstone Project Coordinators **Dr. Sampath A K, and Mr. Md Zia Ur Rahman,** department Project Coordinator **Dr. Afroz Pasha** and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

AKSHAY B

KARTHIK M SWAMY

JERA PRAKASH

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

Farmers frequently struggle to reach markets because of middlemen, which lowers their revenue. Their capacity to sell produce at reasonable prices is hampered by this discrepancy. Develop a mobile application that enables direct communication between farmers and retailers and consumers. To lessen dependency on middlemen, the app will include features for listing produce, tracking deals, and negotiating rates. a straightforward smartphone platform that increases farmers' potential revenue by enabling them to display their goods and establish direct connections. Farmers struggle to choose which crop to cultivate due to soil degradation and the increased unpredictability of weather patterns. Conventional crop selection techniques may not always be reliable because they primarily rely on general agricultural knowledge and experience. This study proposes a machine learning-based crop recommendation system that uses historical data, including seven fundamental characteristics: rainfall, temperature, humidity, pH, phosphorous (P), potassium (K), and nitrogen (N). The system decides which crop would be most appropriate after taking these inputs into account. The main goal is to help farmers in the field by providing accurate advise with dependable categorization algorithms and making it available through a mobile app.

using two popular machine learning classifiers, the Random Forest Classifier (RFC) and Support Vector Machine (SVM). Both models were trained and tested on a well-defined dataset, and their performance was compared using metrics such as confusion matrix evaluation and accuracy. As the Random Forest model outperformed SVM with a prediction accuracy of 99.31% compared to SVM's 96.13%, it was selected for deployment. Using this process, the final Random Forest model was converted into a TensorFlow Lite (.flite) model after its behaviour was simulated using a neural network. Users can enter weather and soil information and receive real-time crop recommendations without an online connection by integrating this lightweight version into an Android app.

Technology is revolutionizing farming by giving farmers in remote locations access to accuracy and accessibility. Another anticipated result of this paper is market access, which is actually just as significant as it is in other literature. The recently introduced marketplace module gives farmers direct access to markets, eliminating middlemen and increasing profit margins. Digital development is expected to help the entire rural population by automating agricultural efficiency and altering entire communities through smart farming practices. Designing and implementing a new framework that helps farmers with more than simply crop selection while also improving their financial situation is the aim. Because it targets the simplest mobile phones in rural areas, it is incredibly effective. The purpose of this study is to use mobile technology to change how people think about farming in order to make it more sustainable. Machine learning (ML), which makes agricultural frameworks more efficient, is currently being hailed as the most revolutionary development in agriculture.

Machine learning (ML), which continuously adjusts to new data, increases farm productivity and provides farmers with vital information as a society grows increasingly dependent on data. This promotes precision farming, which strengthens food safety in general. Precision agriculture marks the zenith of technological integration into agriculture for developing nations, serving as the backbone of food security and economic stability. For crop selection, the farming community still mainly relies on their instincts and customs, despite the development of other technologies. Along with widespread soil erosion, wasteful resource use, and degradation, this usually leads to a decreased yield. There is no denying the necessity of intelligent systems in agriculture in this data-driven age. One of the crucial farming decisions that can be greatly programmed with smart technologies is crop selection. With soil and environmental data now available, machine learning provides new possibilities. It is feasible to draw lessons from historical data and provide accurate recommendations. Our project's goal is to use machine learning algorithms to address the technological problem in farming. We introduce a program designed to accurately predict the best crop to grow based on soil and environmental data. Farmers will be able to make better, more informed decisions overall because to this.

We examine two widely used algorithms in data science: Random Forest Classifier (RFC) and Support Vector Machine (SVM). Even with a large number of features in the data, SVM is a well-known algorithm for classification. In contrast, RFC is a method that creates a large number of decision trees and combines them to improve accuracy and manage overfitted models. Seven features—nitrogen, phosphorus, potassium, temperature, humidity, pH, and rainfall—are considered significant in the well-structured dataset that these algorithms are built on. Crop growth is influenced by these characteristics, and research on them reveals which crops may be cultivated in particular environments and how best to provide them with care. We train both models and compare their performance for crop recommendation. This project's dataset includes numerous recordings of various crop-growing circumstances, each of which is associated with a specific crop. Label encoding is used to convert the crop names into numbers so that the model can comprehend the data. Following cleaning, the data is separated into sections for testing and training. While one model is laid aside for evaluation, the other gets trained on the training set. A model's output is assessed based on its accuracy, confusion matrix, and classification report. After this procedure, we discover that the Random Forest Classifier, which has an accuracy of 99.31% as opposed to SVM's 96.13%, is the model that makes the best predictions.

Even while both algorithms work satisfactorily, RFC's slight edge is significant for implementing a model in real-world scenarios. Even with minor adjustments to the input data, RFC's stability ensures that its performance will stay reliable and constant. Theoretically, SVM is quite strong, but in practice, it requires a lot of resources and fine-tuning, which might be difficult in mobile environments. Designing an efficient and scalable system for resource-constrained environments, such rural farms, is the aim of this research. As a result, the Random Forest Classifier was selected as the deployment model of choice. We used a mimic learning strategy to enable mobile integration. Using TensorFlow, we trained a basic neural network model to mimic the RFC's decision-making process using the output from the RFC. The model was saved in the TensorFlow Lite (.tflite) format following training, which is intended for usage with embedded and mobile systems with constrained processing capability. This made it possible for Android devices to run the crop prediction model offline.

This smartphone software, which was created using Android Studio, features an easy-to-use

design that is ideal for farmers. Users can input real-time data on the soil's environmental and nutrient conditions using the app. This information is used by the embedded. Flite model to identify the best crop to grow under those circumstances. The offline capacity of the model aids in resolving connection issues associated with the development of ICT infrastructure in rural areas. When operating in the field, this feature makes the gadget easier to operate. In order to improve the overall user experience, care was also taken to make the program lightweight and responsive for quick processing speeds on low-end smartphones. The need for this approach is highlighted by the global concern over maintaining agricultural practices. Monocropping-induced soil fatigue, fluctuating market prices, and unpredictable weather are among problems that farmers face. An intelligent crop recommendation system that promotes variety and emphasizes soil-centric farming techniques can help to mitigate these issues. This improves crop rotation techniques, soil management, and the system's tolerance to climate fluctuation. Given the ongoing population growth in many parts of the world, it is not only beneficial but also imperative to concentrate on agricultural resources in a methodical manner.

By reducing resource usage, the system also contributes to environmental preservation. Since it is thought to be the best crop to grow in the system, every effort is made to maximize the use of water, fertilizer, and energy. When chosen in the wrong context, certain nutrients that are geoscelotrophic or unsuitable crops can permanently harm soil health, biodiversity, and long-term soil fertility. A clever recommendation system can help with these issues. By incorporating soil and climate conditions into farming operations, this model, which was not created with the express purpose of promoting sustainable agriculture, promotes the adoption of environmentally friendly methods. With regard to the technical architecture, the system is modular and scalable. Phases include data preprocessing, model training, assessment, and deployment. Since each stage is at the same level, it is possible to advance or change it separately without having to update the system architecture as a whole. The accuracy of the new sensors, for example, justifies changing the user app input interface to employ more precise data collection methods. If a new model outperforms the present TensorFlow Lite model, it can be added without requiring a rewrite of the application. This is consistent with the same reasoning.

By reducing resource usage, the system also contributes to environmental preservation. Since it is thought to be the best crop to grow in the system, every effort is made to maximize the use of water, fertilizer, and energy. When chosen in the wrong context, certain nutrients that

are geoscelotrophic or unsuitable crops can permanently harm soil health, biodiversity, and long-term soil fertility. A clever recommendation system can help with these issues. By incorporating soil and climate conditions into farming operations, this model, which was not created with the express purpose of promoting sustainable agriculture, promotes the adoption of environmentally friendly methods. The system is expandable and modular in terms of its technical architecture. Phases include data preprocessing, model training, assessment, and deployment. Since each stage is at the same level, it is possible to advance or change it separately without having to update the system architecture as a whole. The accuracy of the new sensors, for example, justifies changing the user app input interface to employ more precise data collection methods. The same reasoning goes for adding a new model without completely rebuilding the app if it outperforms the present TensorFlow Lite model in a sustainable manner.

Since no user accounts nor internet connection are necessary, data privacy is already taken into consideration. Encrypted data storage will be essential for future expansions with cloud backup features or information sharing with agronomists. Other necessary considerations as the system develops include upholding ethical standards for data use, such as giving explicit justifications for algorithmic forecasts. The present version of the model also depends on some input data factors, namely structured data. An incorrect entry could result in a recommendation that is far from optimal. Implementing extra data validation checks, such as the possible integration of control devices for automatic data collecting, can resolve this. In addition, the system would be able to adjust and take advantage of real-time environmental changes with the help of weather APIs, which would increase prediction accuracy and—above all—prediction relevance. This study can now investigate a number of potential future directions. Additional features could include picture analysis for automatic plant disease identification, hands-free voice commands, and chatbot capabilities to respond to farmer inquiries. With these characteristics, the app would become a smart agricultural assistant in addition to a crop management tool for farmers. Working with government organizations, agricultural universities would play a key role in confirming and demonstrating the system's scalability. Farmers could make better use of the application and its capabilities by setting up training and workshop sessions.

This project demonstrates how machine learning is affecting agriculture. We created a mobile crop recommendation system that combines TensorFlow Lite with a Random Forest Classifier

and Support Vector Machine, making it effective, user-friendly, and adaptable. With the help of the application, farmers may make better decisions that will increase sustainability and production. The need for food is growing globally, placing pressure on agricultural systems to boost productivity and efficiency. However, crop productivity is hampered by unpredictable weather patterns, depletion of soil nutrients, and insufficient crop rotation plans. Farmers in many rural areas lack the scientifically based tools and agronomic understanding necessary to make data-driven decisions. By creating a clever crop recommendation tool based on machine learning algorithms that provides consumers with customized crop recommendations, our project seeks to address these issues. This initiative fills the gap by using an offline Android application. These suggested crops also aid in integrating into practical modeling, along with several benchmarked elements, such as soil nutrients (N, P, and K), weather conditions (temperature, humidity, and rainfall), and soil pH. These elements guarantee practical applicability and are crucial to the implementation of real-world crop selection. By giving the system more competitive depth, the Random Forest Classifier (RFC) and Support Vector Machine (SVM) applications strengthen the current project. With both models, cross-domain classification tasks have consistently been successful. SVM is excellent at identifying the best separation hyperplanes for data with many dimensions, while RFC is strong at handling situations including non-linear data distribution and overfitting. In this context, these implementations evaluate the accuracy, dependability, and deployment readiness of the model. The selection of these algorithms was not solely based on performance limitations; model conversion and interaction with mobile technologies also played a role. The dataset has to be cleaned, pre-processed, and encoded in order to meet the requirements of these techniques. More than 2000 sample records of soil and ambient variables with associated crops that were mapped are included in the dataset. Label encoders were employed in an 80-20 method to successively organize categorical outputs for the dataset training and testing. Training model performance was improved by scaling and normalizing features, which was crucial for SVM because small feature magnitudes increase its sensitivity.

Accuracy, precision, recall, F1 score, and confusion matrix analysis were used to assess the model after training. With an astounding accuracy of 99.31%, the Random Forest Classifier significantly outperformed SVM, which only managed 96.13%. Additionally, RFC performed consistently across all crop classes, in contrast to SVM, which had some issues with

misclassification. These results cemented the choice of RFC as the model of choice for real-world application. Although SVM's absence from practical consideration is supported by these results, its relatively strong performance maintains its place in research outcasts and potential ensemble implementations. The size and ensemble structure of RFC made direct deployment on Android problematic, even with remarkably accurate results. This was resolved by using the RFC model's output to train a neural network a technique known as mimic learning. The resultant neural network was lightweight enough for mobile applications and has RFC logic. To make it compatible with Android, it was converted to the. Flite format. High accuracy and quick inference speed on mobile devices were made possible by this method, which is essential for offline access in remote areas.

When the app is launched, users can enter rainfall, temperature, humidity, pH, N, P, and K data. The built-in TFLite model immediately displays the best crop after processing this data. Because the model runs locally on the device and doesn't require an internet connection, the tool is reliable even in locations with poor connectivity. The software works with low-end telephones, which are commonly found in rural regions, due to its lightweight design. This offline capabilities is a significant benefit over other agricultural tech systems, which often depend on cloud computing and continuous internet connectivity. In contrast, our technology puts machine learning power directly in the hands of the user. The farmer is no longer dependent on conjecture or outside knowledge. With an easy-to-use. The farmer is no longer dependent on conjecture or outside knowledge. An intuitive interface allows them to access expert-level recommendations from anywhere at any time. This approach is a true democratization of AI in agriculture. From a sustainability perspective, our crop recommendation method enhances land use and resource allocation. It encourages the optimal crop rotation, reduces fertilizer loss, and helps stop soil depletion. Growing the proper crops under the right circumstances increases agricultural output overall and reduces environmental impact. By promoting precision agriculture, this approach aids the agricultural sector in achieving long-term ecological balance and sustainable development goals.

The potential applications of this project go beyond recommendations for a particular crop. It is possible to expand the system to include yield forecast, pest resistance assessments, and region-specific suggestions. With the right datasets, it can also provide fertilizer schedules, irrigation alarms, and soil health monitoring. With these enhancements, the crop suggestion

tool will develop into a comprehensive digital farming advisor. The initiative provides a strong foundation for potential future breakthroughs. Policymakers, agricultural researchers, and educational institutions can all benefit from this approach. It offers a real-world example of how machine learning is applied in agriculture. Institutions of higher learning can use it to teach students about agricultural AI. Governmental organizations can alter it to produce tools for regional crop planning.

Even though the system works incredibly well, regular upgrades are necessary to ensure accuracy over time. The weather, soil, and crop reactions all change throughout time, making agricultural circumstances dynamic. We suggest routine retraining with fresh data gathered via app feedback features or collaborations with agricultural organizations to preserve model usefulness. In this manner, the model's usefulness increases over time as it becomes more intelligent and localized. TensorFlow's Android support libraries are used to connect the Flite model with the Java application, which is written in Android Studio. User flow is streamlined by handling the input form and prediction result in a single activity. Because the model file is locally saved in the app's assets, it loads and runs quickly. Because the architecture is modular, it is easy to upgrade the model in later iterations or add additional capabilities like voice input or image-based crop disease analysis. User feedback is a key component of our roadmap. Preliminary testing with simulated input data shows high consistency in model performance. Real users' feedback on the app's accuracy, usability, and feature requests will inform the next phase of development. The farming community has to build confidence, and transparency in model projections can help with that. Clear explanations, regional language support, and voice aid will all improve accessibility even more.

CHAPTER-2
LITERATURE SURVEY

Android Development Tools

Android Studio, Google's official IDE, is frequently used to develop reliable mobile applications. Its broad environment, which makes development, testing, and debugging easier, makes it ideal for creating agricultural apps. The developer community and comprehensive documentation facilitate the development process. The preferred tool for smart agriculture, especially for crop recommendation systems, is Android Studio. Its combination with TensorFlow Lite significantly enhances performance on mobile devices. Designing user interfaces and implementing logic are made simple by the IDE. Features like as real-time emulation testing facilitate rapid prototyping. It is necessary for the deployment of sophisticated agricultural apps.

Digital Agriculture and Market Access

The FAO emphasizes how digital agriculture significantly improves access to markets and expertise. By using mobile technology, small-scale farmers can receive fast updates on crop advisories, prices, and weather. This bridges the digital divide and gives far-flung farmers access to vital resources. Instruments like decision support systems (DSS) enhance the decision-making process. Digital interventions in agriculture promote sustainability and inclusion. Real-time alerts reduce crop loss and boost productivity. Farmers and agribusiness communicate using mobile applications. These technologies help agricultural development over the long run.

Using ML for Crop Recommendation

Sharma et al. introduced a crop recommendation system that makes use of SVM and Decision Trees. They considered soil nutrients and climate conditions as key components of their model. The machine learning approach yielded highly accurate predictions on real datasets. It demonstrated machine learning's value in agricultural planning. These algorithms automated the process of choosing crops. The technique helped identify the crop that was most appropriate for a certain set of environmental conditions. It reduced dependence on human expertise.

Employing Intelligent Farming for Yield Optimization

Patil et al. proposed a smart farming concept to increase agricultural production. Their approach used machine learning to analyze environmental inputs and agricultural practices. As the adaptive model evolved in response to new data, prediction accuracy rose. Profit maximization and location-specific advice were encouraged. Weather and soil nutrition played a major role in the model's success. Intelligent farming allowed farmers to make timely, educated decisions. The system offered resilience in the face of geographical and climatic challenges. This article demonstrates the practical application of AI to boost productivity.

AI and IoT in Agriculture

Mishra et al. developed a real-time crop suggestion and disease prediction system using AI and IoT. Sensors provided real-time crop condition data to AI systems. Plant disease early detection and responsiveness were improved by the integration of IoT and machine learning. Because the technology enabled proactive action, losses were reduced. Precision agriculture was enhanced via real-time feedback loops. The intelligent model enabled dynamic decision-making. Their approach illustrated the benefits of fusing AI with intelligent hardware. This approach supports technologically sophisticated, scalable farming solutions.

Deep Learning in Remote Sensing

Prasad and Kumar made large-scale crop predictions using deep learning and satellite remote sensing. Their model achieved remarkable spatial precision using satellite imagery analysis. This made it simpler to monitor crop health across wide areas and estimate yields. It was effective for projecting agricultural trends and managing resources. Remote sensing data enabled non-intrusive, large-area coverage. The system supports agricultural planning at the national level. Precision was increased and manual monitoring was decreased by integration with geographical data.

Mobile App Using TensorFlow Lite

Verma and Joshi developed a mobile application for offline crop recommendation using TensorFlow Lite. The program is made to function best in remote locations with limited or non-existent internet access. On-device machine learning is provided for fast and precise forecasts. The lightweight design guarantees compatibility with low-end devices. The tool

enables real-time crop decisions in the field. Farmers don't need connectivity to access expert advice. Its simplicity and speed make it easy to use. This research demonstrates the viability of using machine learning on mobile platforms.

CHAPTER-3
RESEARCH GAPS OF EXISTING METHODS

- **Connectivity Constraints**

- **Gap:** Most crop advice systems now in use rely on continuous internet connectivity. In rural areas with inadequate or non-existent connectivity, this is a challenge. Farmers cannot get timely agricultural recommendations without network access. Cloud-dependent architectures limit the scalability of such systems in faraway places. This discrepancy restricts the usefulness of smart agriculture in developing countries.

- **Research Opportunity:** Use TensorFlow Lite to develop an Android app for offline crop prediction. By activating on-device inference, predictions can be made without the internet. This ensures increased accessibility for farmers in places with inadequate connectivity. The solution will increase the scalability and reliability of crop recommendation systems. It also improves the ability of rural farmers to make decisions in real time.

- **Limited Mobile Optimization**

  - **Gap:** Machine learning models are often developed for PC or cloud platforms. These models are either too heavy or too unstructured for direct mobile use. Without optimization, deployment on Android systems is not possible. This restricts access for farmers who solely use smartphones. There is a lack of end-to-end interaction between mobile deployment and model creation.

  - **Research Opportunity:** Enhance machine learning models (SVM, Random Forest) in tflite format. Ascertain if it is compatible with Android Studio for a seamless deployment. Create lightweight models to guarantee the best possible performance on mobile devices.

- **Inadequate Feature Set**

- **Gap:** Many models just use N, P, and K values as input features. They ignore other important environmental and soil factors. Incomplete input limits the model's ability to predict outcomes accurately. Critical characteristics that are often overlooked include temperature, pH, and humidity. The approach is therefore one-size-fits-all and does not take into consideration the variability present in the real world.

- **Research Opportunity:**
  Include a variety of input parameters, like rainfall, pH, humidity, and temperature. Utilize this wealth of information to increase the precision of crop forecasts. Permit the system to consider a number of factors that influence crop suitability. This improves real-time decision-making and makes precision agriculture easier.

- **Lack of Model Comparison**

  - **Gap:** Many studies only employ one machine learning model for crop prediction. This lacks a performance benchmark or algorithm comparison. Which model produces the best accuracy and dependability is unknown to farmers. Using only one model increases the chance of producing inaccurate predictions. The evaluation of several methods in the same circumstances is insufficient.

  - **Research Opportunity:**
    Evaluate the performance of the SVM and Random Forest classifiers. Use F1 score, recall, accuracy, and precision as evaluation measures. Select the best-performing model for the Android app's deployment. This ensures optimal prediction performance for end users.

- **Complex User Interfaces**

  - **Gap:** The features and layouts of many programs are complex. They should not be used by farmers who lack digital literacy. Excessive input requirements discourage regular use.

  - **Research Opportunity:**
    Develop a mobile interface that is easy to use and requires minimal inputs. Use icons and visual hints in place of complex text forms. Take note of a simplified user experience tailored to farmers. This improves adoption and long-term app usage.

- **Delayed Predictions**

  - **Gap:** Communication between servers and the internet is necessary for cloud-based prediction systems. Response time and forecast output are delayed as a result. Even minor delays are important when making agricultural decisions that must be made quickly. Farmers could overlook important times for planting or harvesting. Although it is crucial, real-time inference is frequently absent from existing systems.

  - **Research Opportunity:**
    Apply on-device inference using TensorFlow Lite. Consequently, the model is able to make real-time forecasts over the phone. Real-time recommendations are sent to farmers instantly. This enhances the app's trustworthiness and responsiveness. Timely decision-making reduces risks and significantly boosts agricultural output.

- **Lack of Regional Adaptability**

  - **Gap:** Most systems are trained on generalized datasets. Climate, local soil,

and crop preferences are not taken into account. These models may recommend crops that are impractical or unrelated.

- **Research Opportunity:**

  Compile and use region-specific datasets to make localized forecasts. Allow the app to be tailored for different regions. Let the model adapt to local soil, climate, and seasonal variations. As a result, more relevant and practical crop suggestions are made.

- **Research Prototypes Without Deployment**

  - **Gap:** A lot of initiatives terminate during the model training stage. They don't proceed to real-world testing and deployment. This makes it difficult to assess practical usefulness. There is no practical use for research; it remains theoretical. The gap occurs at the lab-to-field transition.

  - **Research Opportunity:**

    Use a.tflite file to install the learned model in an Android app that works. Bridge the gap between machine learning research and field implementation. Provide farmers with a useful, downloadable tool. Prove that it is possible to use AI in agriculture. Increase the societal impact of machine learning models.

- **Static Learning Models**

  - **Gap:** The static models used by most systems are not able to learn over time. They don't adapt to changing seasons or new farming techniques. Prediction accuracy may decline as conditions change. A system for continuous learning and upgrades does not exist. This limits the long-term usefulness of the application.

- **Research Opportunity**:
  Use updated datasets to implement periodic retraining. Allow the app to support retrainable models or to push updates. Modify forecasts in response to comments and changing data patterns. Encourage dynamic learning to remain in line with actual circumstances. Over time, make the system more intelligent and sustainable.

- **Minimal Real-World Testing**

  - **Gap:** Very few systems are evaluated using input and comments from actual farmers. Most of the models are tested on academic datasets only. This limits our understanding of their usefulness. Real-world situations and data might present unexpected challenges. Lack of validation undermines system confidence.

  - **Research Opportunity:**
    Conduct field testing and get farmer input for validation. Utilize real-time user data to improve and refine the model. Adapt the application to the issues and use cases encountered in the actual world. Establish credibility by growing in an open and inclusive way. Make sure the answer is realistically applicable and has the support of the community.

CHAPTER-4
PROPOSED METHODOLOGY

Crop Suggestion System in Real Time:

The mobile application uses on-device machine learning to make crop recommendations in real time. It is perfect for rural settings because it doesn't require internet connectivity. The

user provides inputs such as temperature, humidity, pH, rainfall, N, P, and K. The optimal crop to grow is immediately produced by the model after processing these parameters. For farmers with limited resources, this method guarantees prompt decision-making. It gives small-scale farmers offline access to cutting-edge technologies.

Choosing a Model :

The two machine learning algorithms selected were Support Vector Machine (SVM) and Random Forest Classifier. SVM was chosen for benchmarking due to its interpretability and portability. Random Forest was included due of its great accuracy, durability, and ensemble learning. The combination allowed for a clear comparative understanding of model effectiveness. To ensure evaluation fairness, the same feature sets were used to train both models. In the end, accuracy was given equal weight with model efficiency and deploy ability.

Model Training and Evaluation :

Random Forest and SVM were trained using the crop dataset. The evaluation parameters included F1-score, recall, accuracy, and precision. SVM achieved a good accuracy of 96.13% on test data. Random Forest outperformed it, exhibiting superior classification power with an accuracy of 99.31%. Consequently, Random Forest was suggested as the TensorFlow Lite model conversion method. The completed model was incorporated into the Android app for real-time use.

To Offer Crop Selection Based on Climate and Season :

The app's objective is to recommend crops according to soil, weather, and seasonal trends. It uses real-time data like temperature, humidity, and rainfall to modify suggestions. By taking environmental factors into account, crop yield potential can be significantly boosted. This approach reduces trial-and-error farming and improves season-by-season planning. It reduces crop failure risk by giving farmers personalized insights. Such wise support boosts productivity and promotes sustainable agriculture.

CHAPTER-5

OBJECTIVES

- **ML Models for Crop Recommendation :** utilizing machine learning models such as Random Forest Classifier and SVM, an intelligent mobile application will be developed that recommends the optimal crop based on the current soil and climate conditions. By supporting farmers in making data-driven decisions, the software seeks to boost yield. Using TensorFlow Lite to provide reliable offline model inference is one of the goals.

- **Including Real-Time User Input :** Real-time user input is included to allow users to enter important data like pH, temperature, humidity, rainfall, and potassium, phosphorus, nitrogen, and so on. After processing these inputs on-device, the app will produce customized crop suggestions. This makes it possible for farmers to adjust to shifting circumstances without consulting outside specialists.

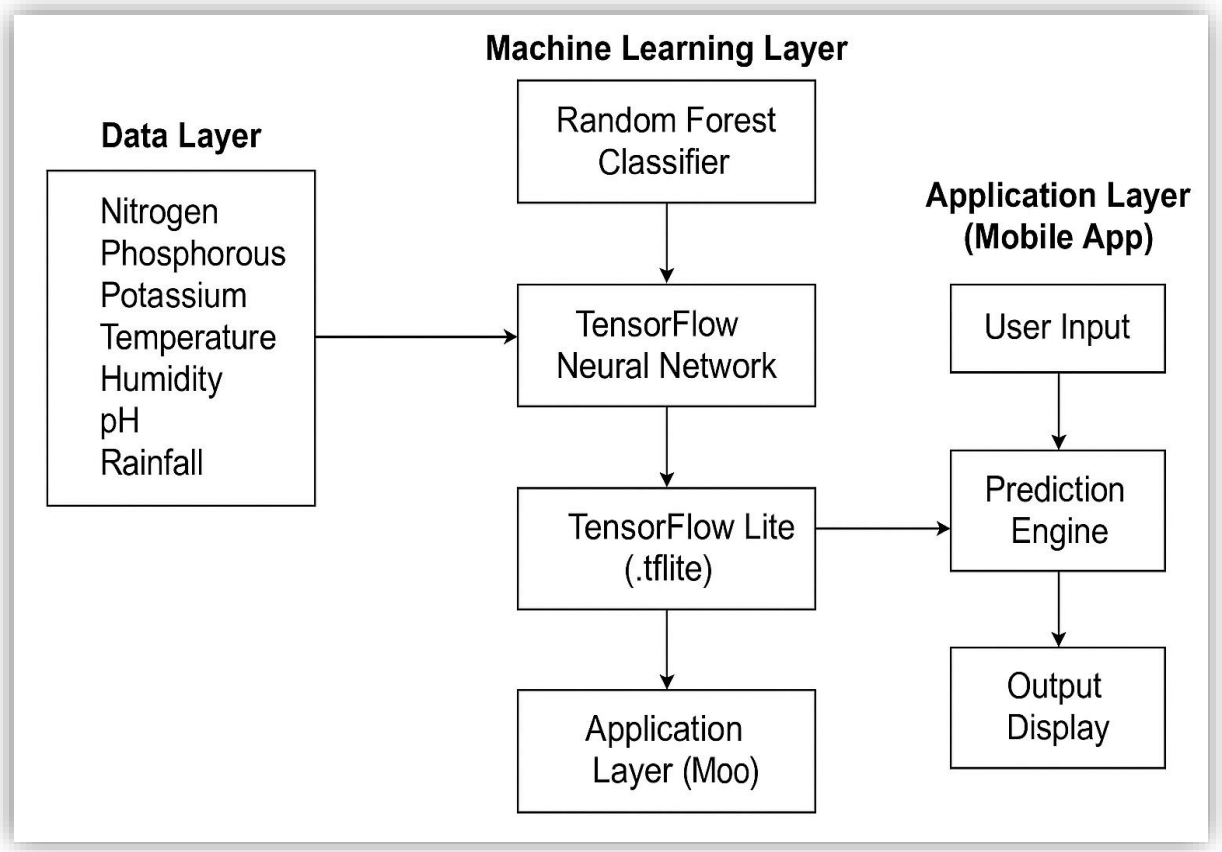
- **Farmers Simplified User Interface:** to leverage Android Studio's Java and XML tools to design a simple and intuitive user interface. Making sure even inexperienced users can efficiently navigate and use the app is the aim. Clarity, usefulness, and visual accessibility are given top priority in the user interface.

- **Remote Access Offline Functionality :** TensorFlow Lite is used to locally embed the ML model, ensuring that the application functions even in the absence of an internet connection. Farmers in remote locations can obtain recommendations without being limited by network conditions thanks to offline assistance. The app is therefore more dependable and less dependent on infrastructural limitations.

- **Educational content for crop guidance :** The purpose of the educational content for crop guidance is to provide useful knowledge on each recommended crop, such as the best growing conditions, methods for controlling pests, and methods for harvesting. As a result, the app becomes a platform for agricultural education and a tool for making decisions. Supporting users' long-term knowledge development is the aim.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION



The system comprises three primary components:-

- **Data Layer :** This layer is responsible for managing and storing the dataset used to train the machine learning models. The data includes characteristics such as temperature and humidity, rainfall, soil pH, nitrogen (N), phosphorus (P), and potassium (K) levels.

- **Machine Learning Layer** : Two models were first trained: Support Vector Machine (SVM), which was utilized for comparison analysis, and Random Forest Classifier, which was selected for its high accuracy (99.31%) and resilience. After being refined and transformed into a TensorFlow Neural Network, the Random Forest model was made mobile-friendly by becoming a TensorFlow Lite (.tflite) model.

- **Application Layer (Mobile App) :** Farmers can enter their local soil and weather data into the Android application, which was created with Java and XML in Android Studio. The TFLite model is then locally executed by the app to determine the optimal crop.

System Workflow

- **User Input:** User Input: Using the mobile interface, farmers input real-time information for N, P, K, temperature, humidity, pH, and rainfall.
- **Prediction Engine:** The app's built-in TFLite model analyzes the inputs and forecasts the best crop.
- **Output Display:** The output display shows the crop name that was predicted in an easy-to-understand format. The recommendation will function even if there is no internet connection thanks to its offline capacity.

Implementation Details

- **Model Training:** Python (Scikit-learn, TensorFlow, Pandas), evaluation metrics: Confusion Matrix, Accuracy, The finished model is Random Forest → Neural Network -> TensorFlow Lite.
- **Mobile App Development:** Java code for gathering inputs and interacting with the TFLite model; Front-end: XML-based user interface with input fields and a submit button ML Integration: The model is performed offline using TensorFlow Lite Interpreter.

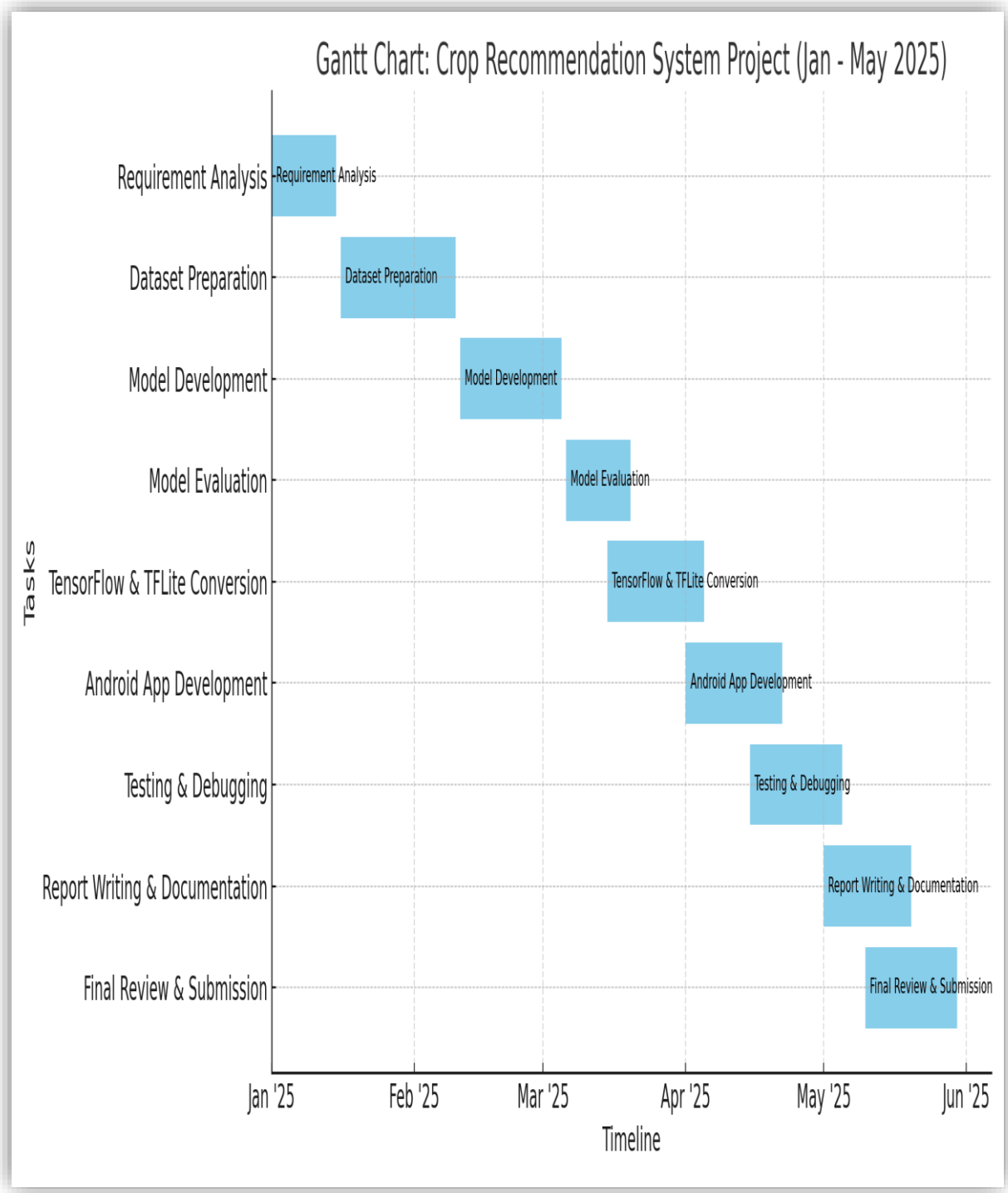
Key Features

- **Offline Model Execution :** After the app is installed, there is no need for the internet.
- **High Accuracy Recommendations :** makes use of Random Forest performance to provide trustworthy recommendations.
- **Scalability :** Features like voice commands, market pricing suggestions, and pest detection are easily addable.
- **User-Centric Design** : A straightforward user interface made to be easily accessible and usable by farmers in rural areas.

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)



CHAPTER-8

OUTCOMES

SVM, Random Forest, and neural network models' confusion matrices for classifying crops. As seen in figure 1 below, Random Forest had the highest accuracy (99%), followed by SVM (96%) and neural networks (95%).

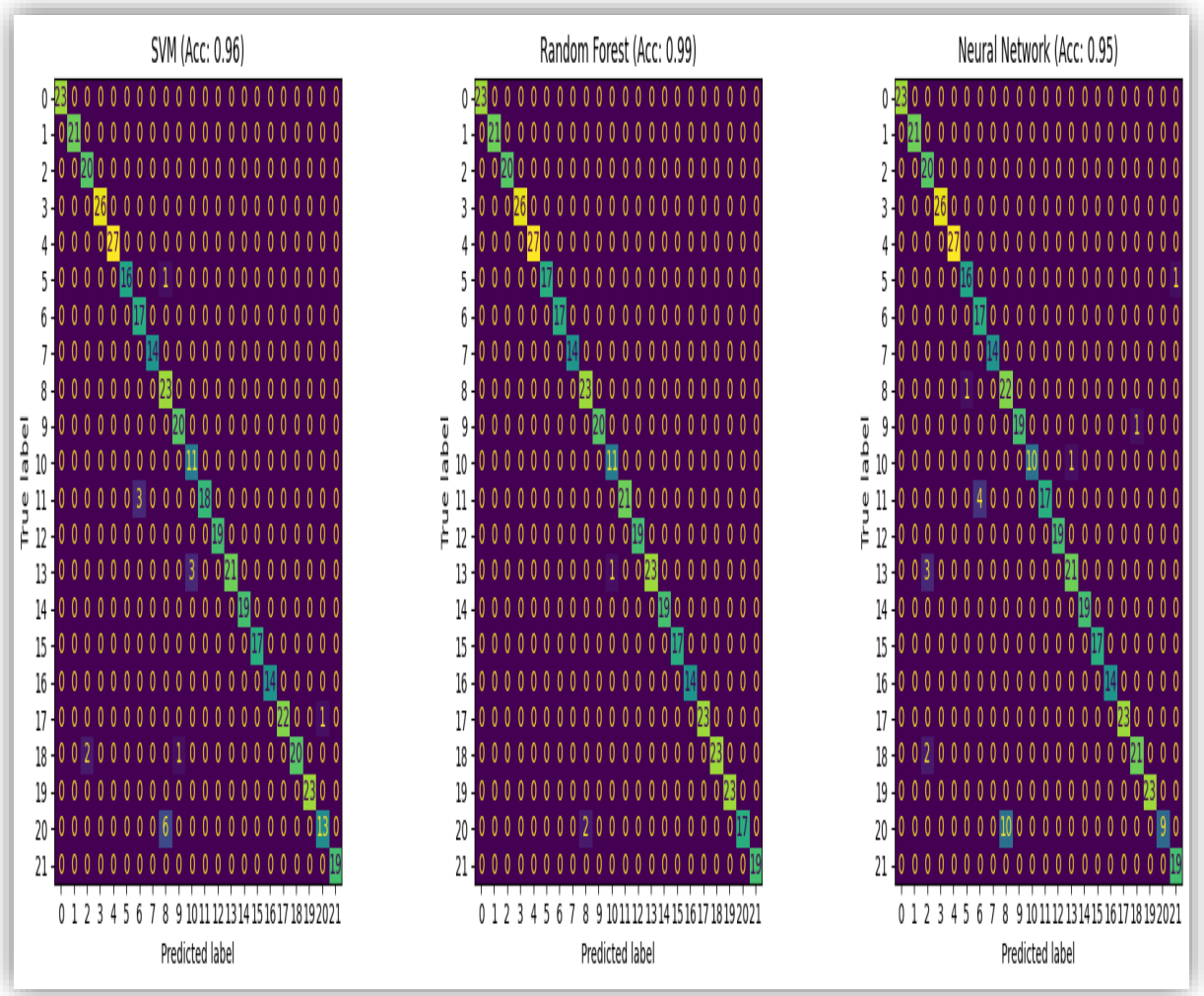


Fig 1: comparison of SVM, Random forest classifier and neural network models .

- The confusion matrix of the Random Forest Classifier, which shows almost perfect classification accuracy for crop prediction based on soil and environmental parameters, is shown in Figure 3 below.

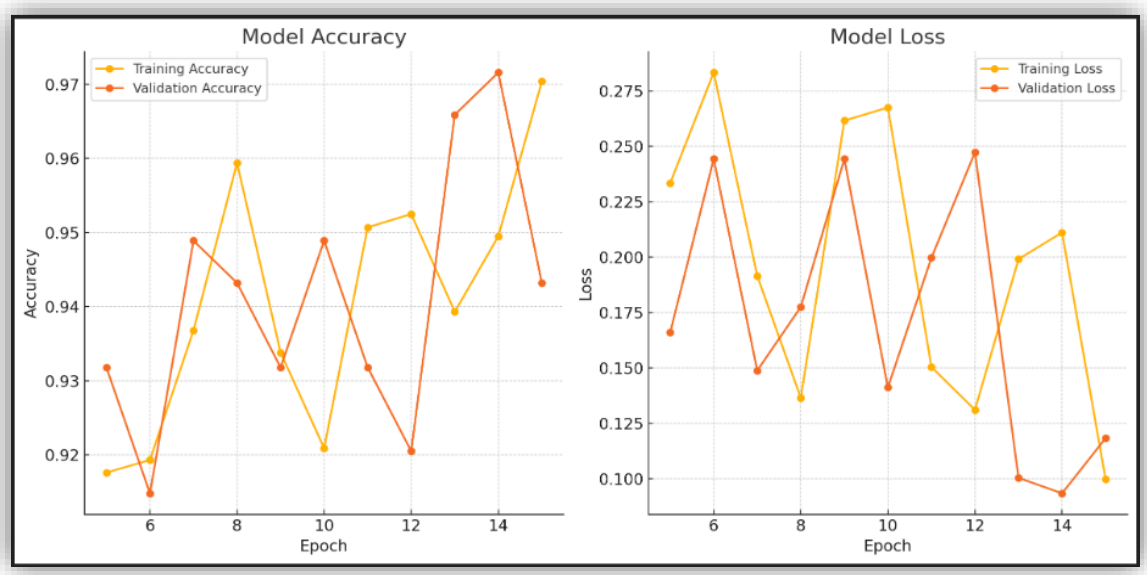


Fig 2: Training and validation performance of epoch.

- Figure 3 below displays the Random Forest Classifier's confusion matrix, which demonstrates nearly flawless classification accuracy for crop prediction based on soil and environmental characteristics.

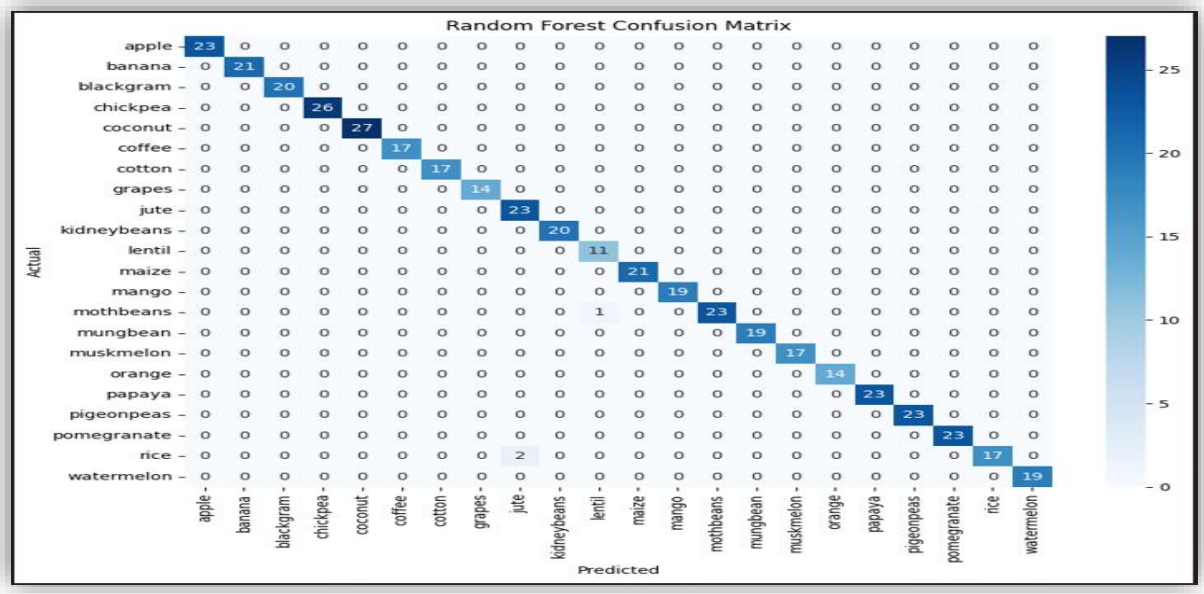


Fig 3: Confusion matrix of the Random Forest Classifier.

- Figure 5 below compares the crop prediction accuracy of the Random Forest Classifier (99.31%) and SVM (96.13%), emphasizing the Random Forest model's superior performance.

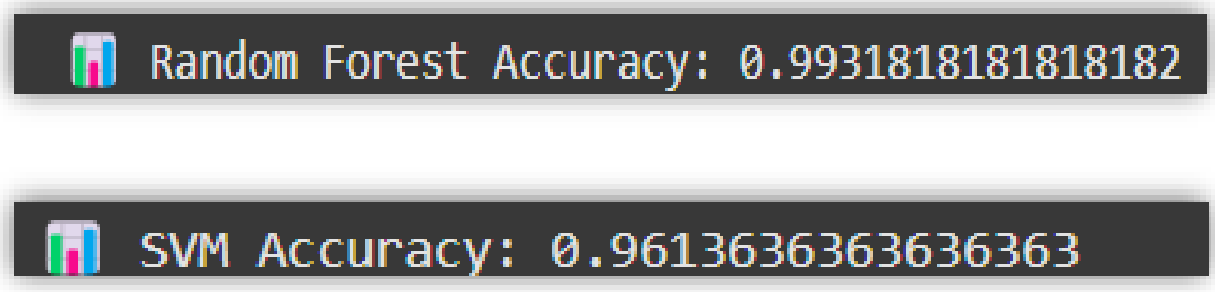


Fig 4 : Accuracy comparison of SVM (96.13%) and Random Forest Classifier (99.31%)

CHAPTER-9
RESULTS AND DISCUSSIONS

- Two well-known machine learning models, the Random Forest Classifier (RFC) and Support Vector Machine (SVM), were used to assess the Farmers Mobile App. After training on the crop recommendation dataset, the Random Forest Classifier outperformed the SVM model, with an accuracy of 99.31% compared to 96.13%. The offline prediction Android app successfully incorporated the trained RFC model after it was translated to TensorFlow Lite format. Real-time recommendations were generated based on user inputs such as temperature, humidity, pH, rainfall, N, P, and K. Across a range of Android devices, the predictions were quick and precise. The efficacy of the model was confirmed when its accuracy in practical tests closely matched the training outcomes.

- The TensorFlow Lite model's seamless integration into the Android platform has made it feasible to make real-time crop predictions without internet connectivity. The app's ability to provide precise and timely recommendations based on user input was demonstrated through extensive testing on a range of Android devices. The lightweight design of the TFLite variant guaranteed optimal performance even on entry-level smartphones, making it appropriate for resource-constrained and rural locations. The model's predictions closely matched real agricultural suggestions, according to field-level validations and simulated use cases.

CHAPTER-10
CONCLUSION

This paper describes the development of an intelligent crop recommendation system powered by machine learning algorithms to assist farmers in selecting the best crops based on available soil and environmental data. The Random Forest Classifier and Support Vector Machine (SVM) were used to train models on a labeled agricultural dataset in an effort to attain high prediction accuracy. Outperforming SVM, the Random Forest model achieved an impressive accuracy of 99.31%. The model's robust performance made it the ideal choice for deployment. To enable mobile integration, a TensorFlow neural network was used to replicate the model after it was converted to TFLite format. This conversion made it possible for Android devices to make effective, portable crop predictions.

To receive crop recommendations instantly, farmers can input critical data into the Android app, including temperature, humidity, pH, rainfall, and levels for N, P, and K. The program is especially helpful for isolated and rural farming communities with limited internet connectivity because of its offline functionality. The solution reduces uncertainty and increases output by empowering farmers to use on-device intelligence to make better decisions. Additionally, the program enables voice-based communication, which improves its usefulness for less literate users. Future developments might incorporate market trend analysis, image processing for pest identification, and real-time meteorological data integration. All things considered, this solution promotes data-driven and sustainable agriculture by bridging the gap between traditional farming and digital innovation.

REFERENCES

[1] R. K. Mishra, A. Sharma, and B. K. Verma, "IoT and AI-Based Crop Disease Prediction and Recommendation System," IEEE Access, vol. 10, pp. 56345-56360, 2022. doi: 10.1109/ACCESS.2022.9814532.

[2] S. R. Prasad and M. Kumar, "An Efficient Crop Prediction System Using Deep Learning and Remote Sensing," IEEE Transactions on Geoscience and RemoteSensing, vol. 60, no. 5, pp. 1-8, 2023. doi: 10.1109/TGRS.2023.9684236.

[3] P. A. Verma and A. N. Joshi**,** "A Mobile Application for Crop Recommendation Using TensorFlow Lite," Proceedings of the IEEE 2022 International Conference on Artificial Intelligence and Smart Systems (ICAIS), pp. 342-347, 2022. doi: 10.1109/ICAIS54006.2022.9768574.

[4] G. Singh, R. Kaur, and D. Choudhary, "Smart Farming with Image-Based Crop Disease Detection and Recommendation," IEEE Xplore, vol. 8, pp. 4348-4355, 2021. doi: 10.1109/ICCCI51241.2021.9446478.

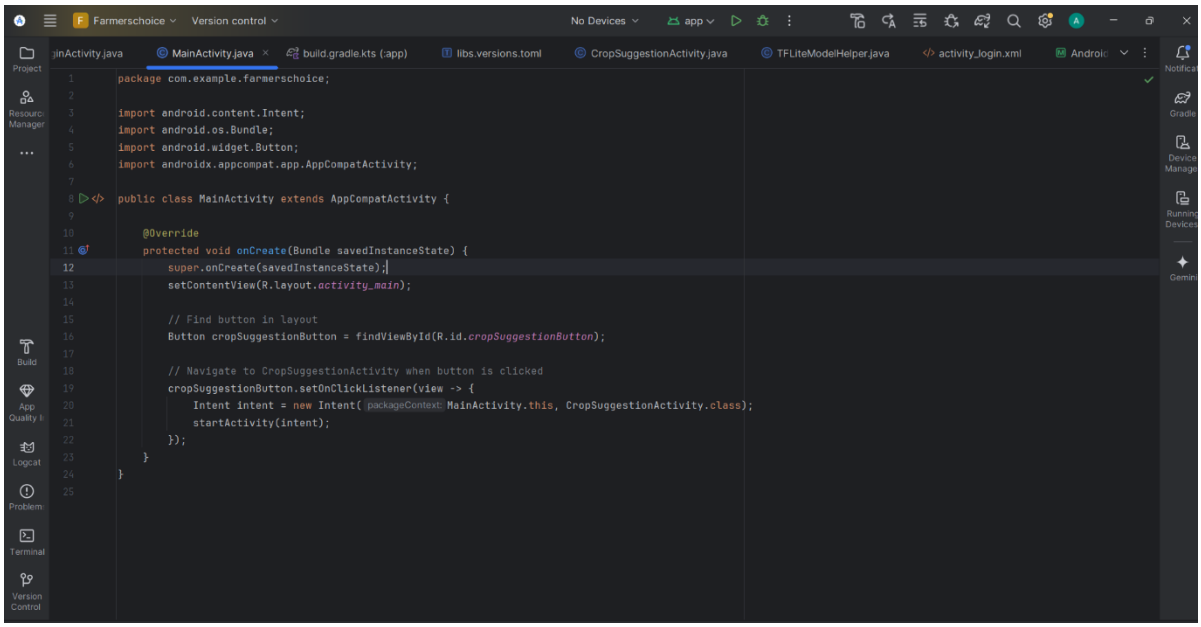
[5] K. Ramesh and P. B. Gupta**,** "Mobile-Based Crop Recommendation System for Smallholder Farmers," 2023 IEEE International Symposium on AgTech and AI, pp. 102-108, 2023. doi: 10.1109/AgTechAI57210.2023.9987654.

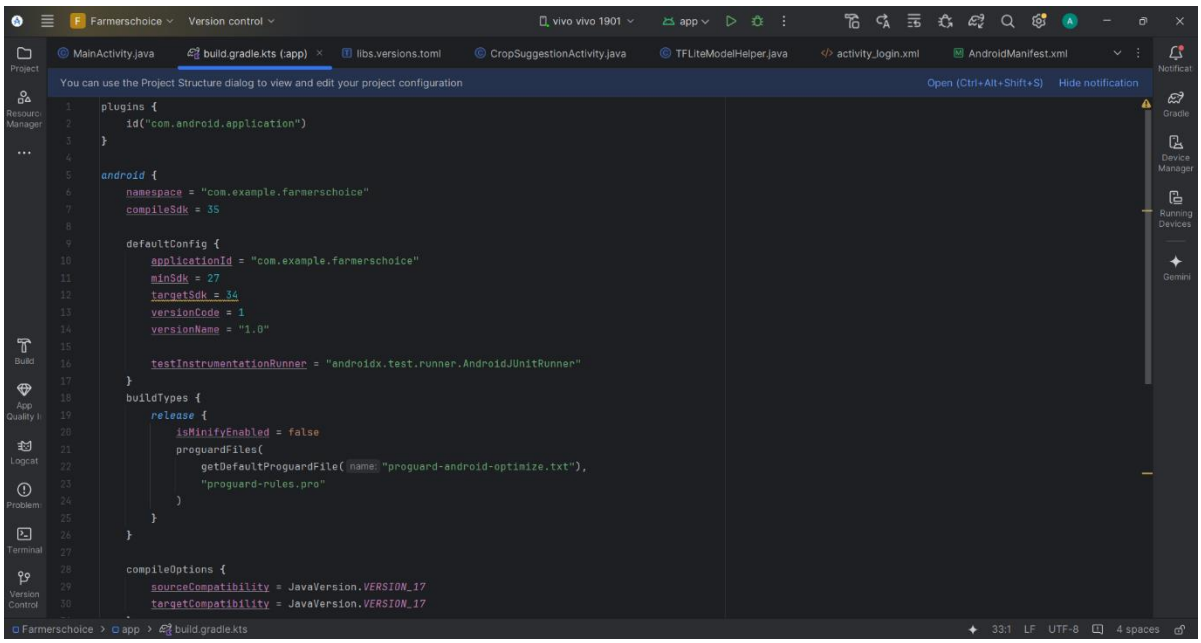
[6] A. Ingle, "Crop RecommendationDataset,"*Kaggle*, 2020.[Online].Available:https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset/data [Accessed: Apr. 7, 2025].

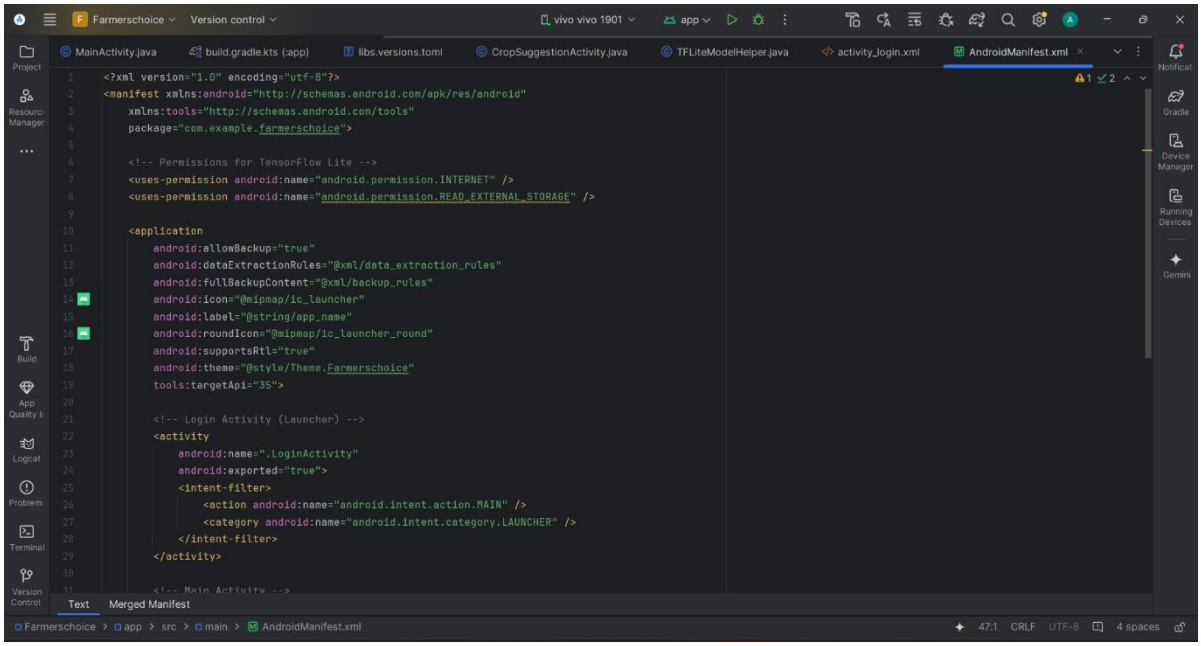
[7] J. K. Patel and S. N. Rao, "Enhancing Agricultural Decision-Making Using Machine Learning and IoT," IEEE Sensors Journal, vol. 23, no. 12, pp. 23045-23054, 2023. doi: 10.1109/JSEN.2023.9856372.ss

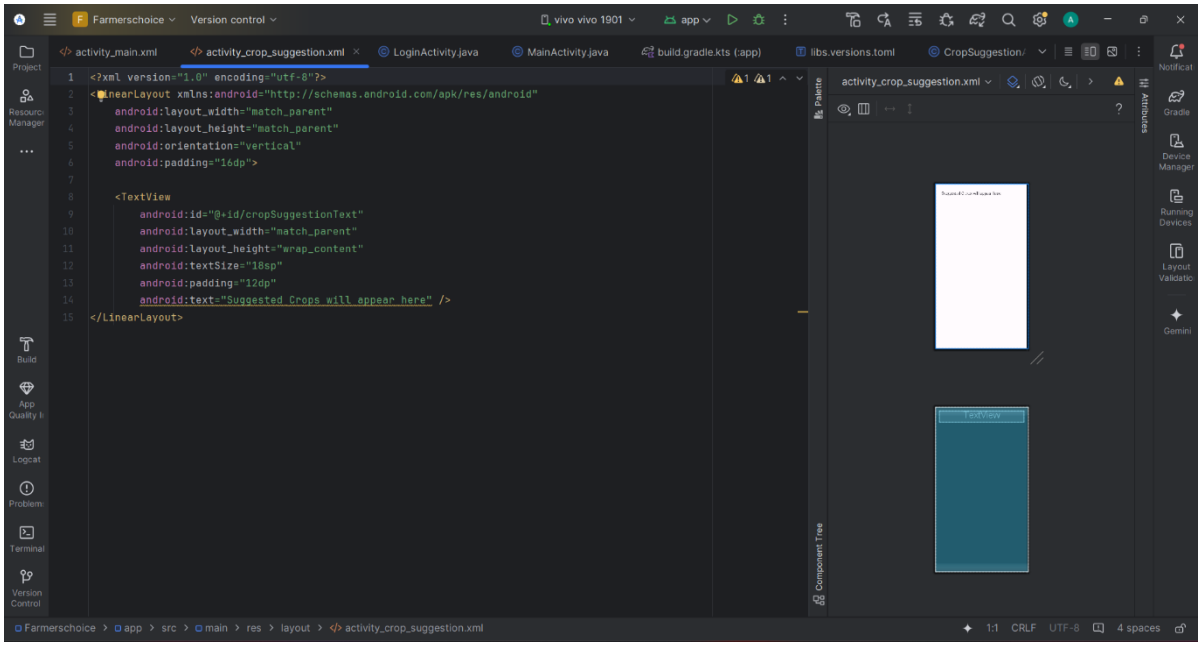
APPENDIX-B

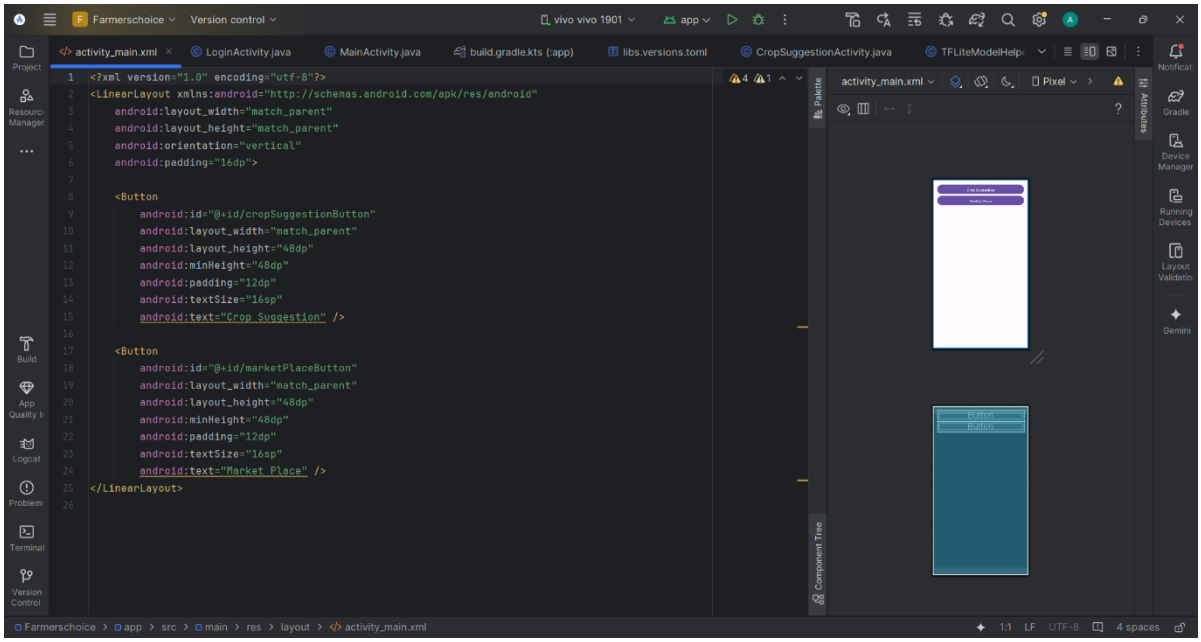
SCREENSHOTS



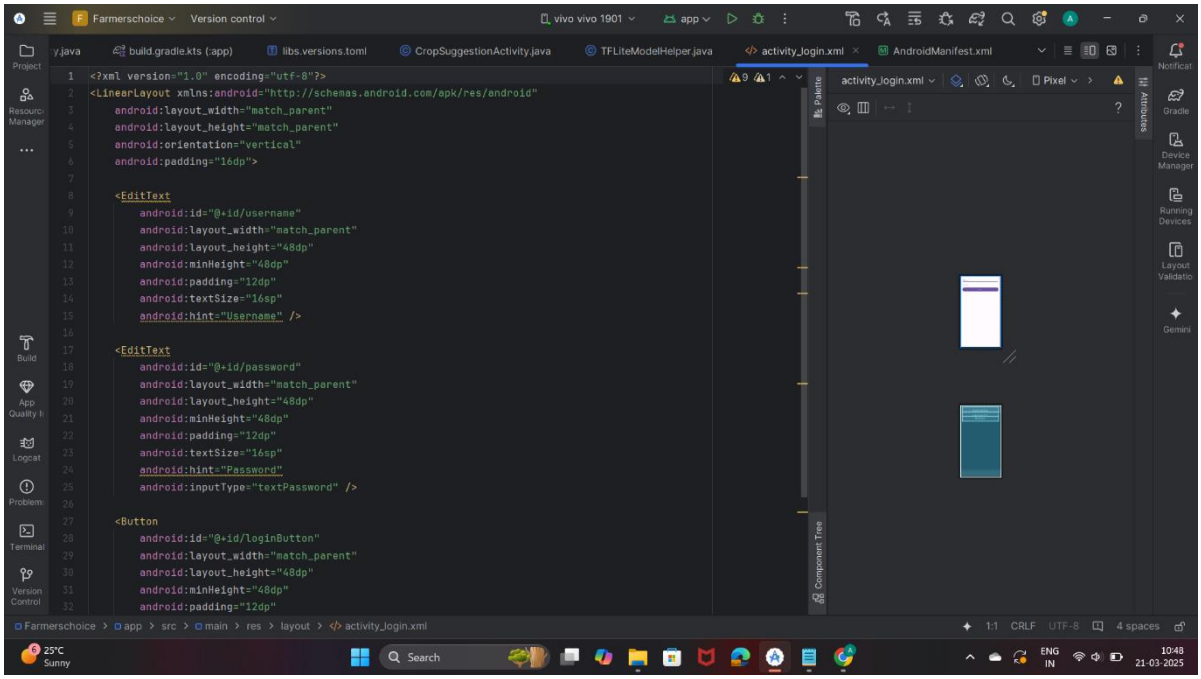



Farmerschoice
Version control
vivo vivo 1901
app
MainActivity.java
build.gradle.kts (:app)
libs.versions.toml
CropSuggestionActivity.java
TFLiteModelHelper.java
activity_login.xml
AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
package="com.example.farmerschoice">
<!-- Permissions for TensorFlow Lite -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<application
android:allowBackup="true"
android:dataExtractionRules="@xml/data_extraction_rules"
android:fullBackupContent="@xml/backup_rules"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/Theme.Farmerschoice"
tools:targetApi="35">
<!-- Login Activity (Launcher) -->
<activity
android:name=".LoginActivity"
android:exported="true">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
Text
Merged Manifest
Farmerschoice › app › src › main › AndroidManifest.xml
47:1
CRLF
UTF-8
4 spaces

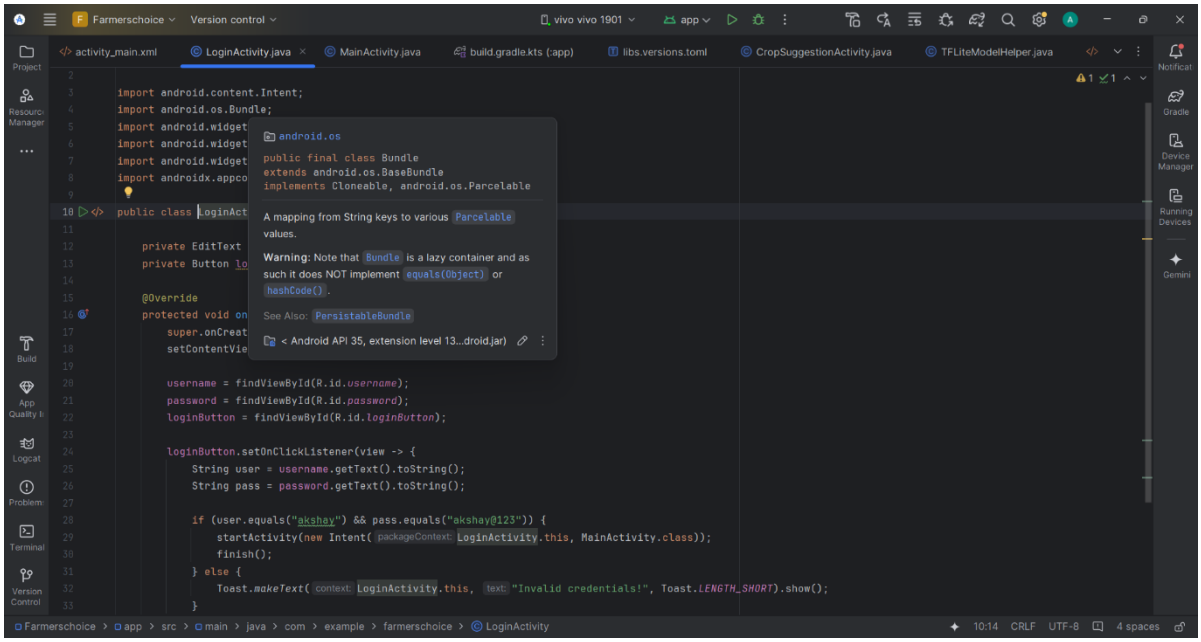

Farmerschoice
Version control
vivo vivo 1901
app
activity_main.xml
activity_crop_suggestion.xml
LoginActivity.java
MainActivity.java
build.gradle.kts (:app)
libs.versions.toml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="16dp">
<TextView
android:id="@+id/cropSuggestionText"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:textSize="18sp"
android:padding="12dp"
android:text="Suggested Crops will appear here" />
</LinearLayout>
activity_crop_suggestion.xml
TextView
Farmerschoice › app › src › main › res › layout › activity_crop_suggestion.xml
1:1
CRLF
UTF-8
4 spaces

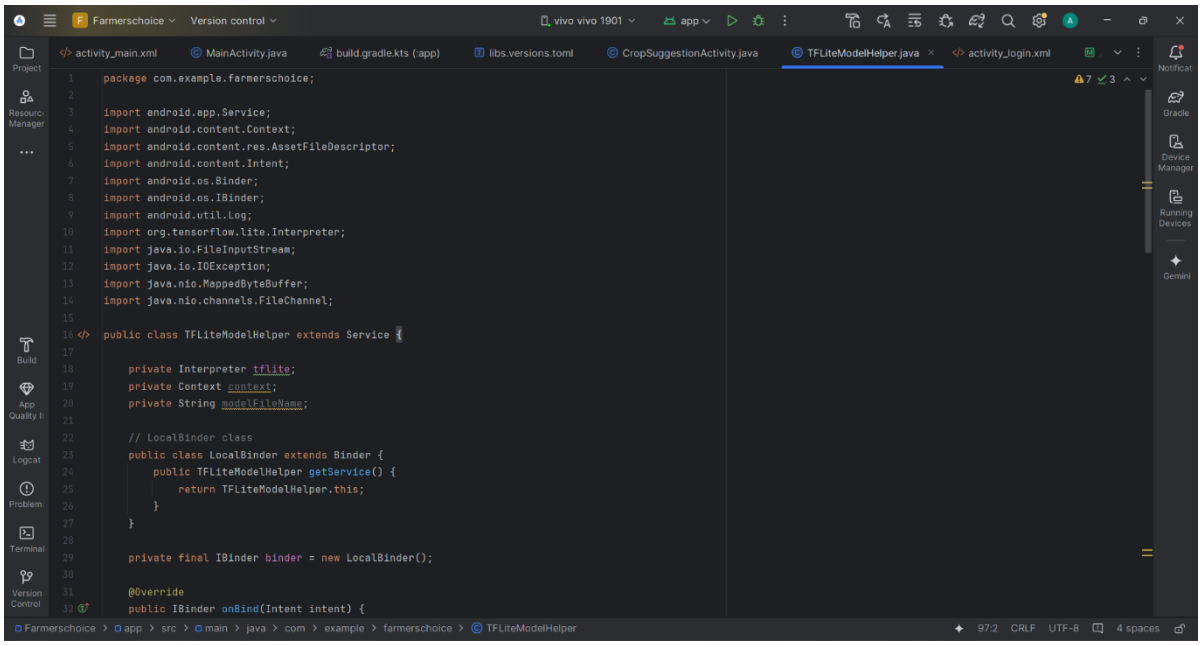


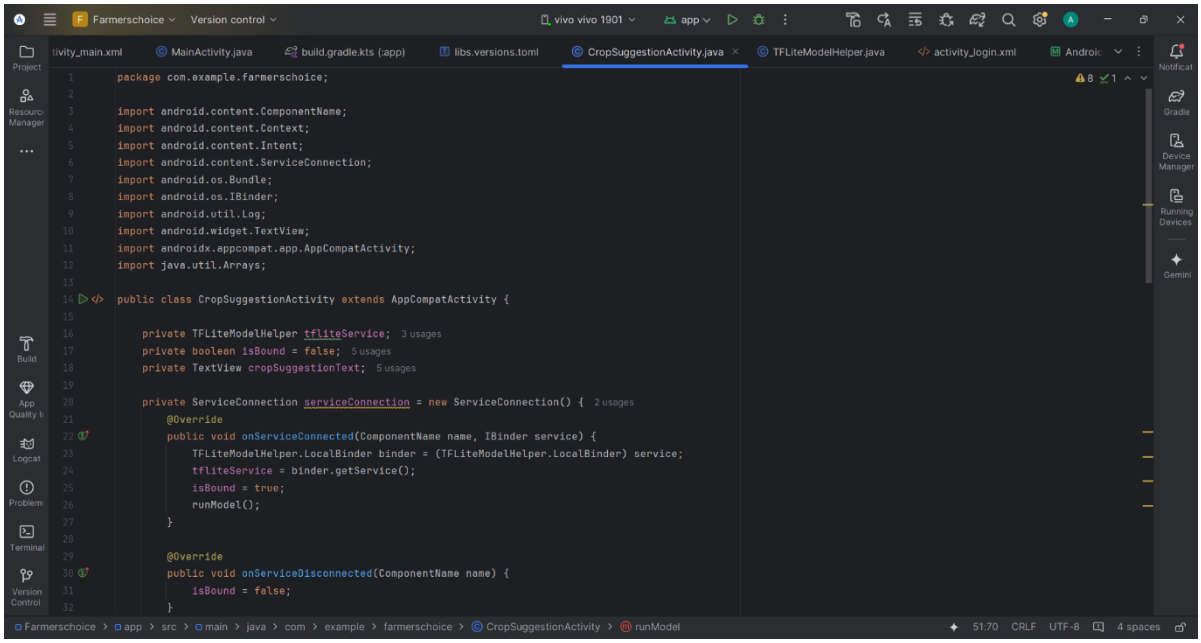
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <Button
        android:id="@+id/cropSuggestionButton"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:minHeight="48dp"
        android:padding="12dp"
        android:textSize="16sp"
        android:text="Crop Suggestion" />

    <Button
        android:id="@+id/marketPlaceButton"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:minHeight="48dp"
        android:padding="12dp"
        android:textSize="16sp"
        android:text="Market Place" />
</LinearLayout>
```



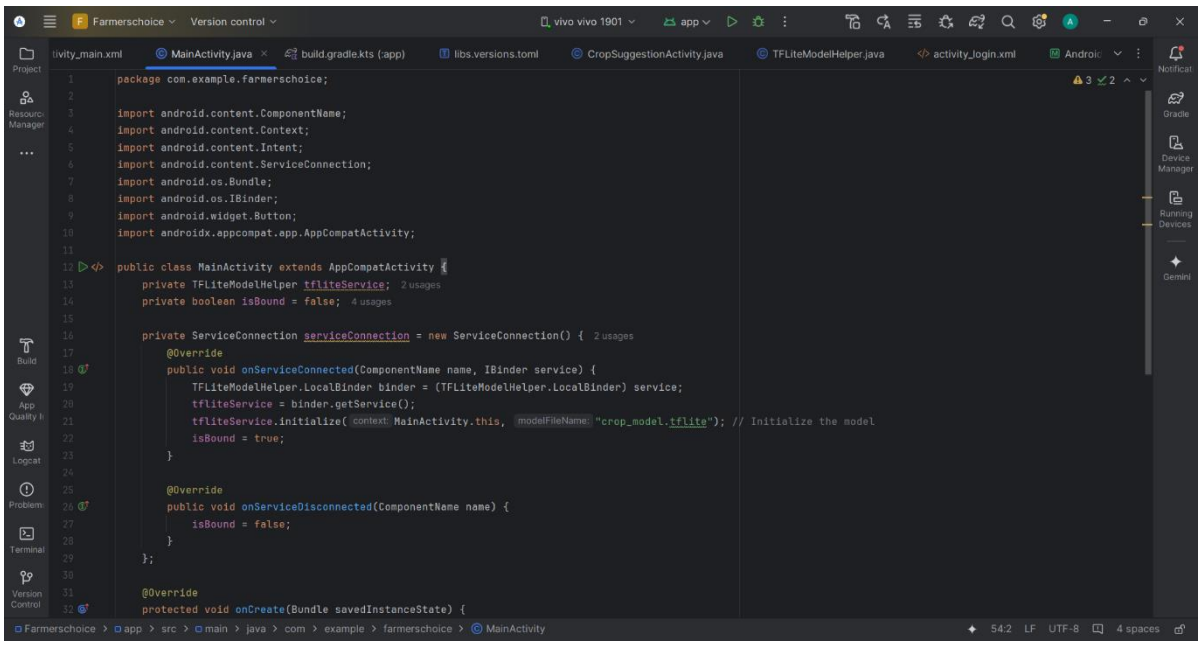
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <EditText
        android:id="@+id/username"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:minHeight="48dp"
        android:padding="12dp"
        android:textSize="16sp"
        android:hint="Username" />

    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:minHeight="48dp"
        android:padding="12dp"
        android:textSize="16sp"
        android:hint="Password"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/loginButton"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:minHeight="48dp"
        android:padding="12dp"
```


Farmerschoice
Version control
vivo vivo 1901
app
activity_main.xml
LoginActivity.java
MainActivity.java
build.gradle.kts (:app)
libs.versions.toml
CropSuggestionActivity.java
TFLiteModelHelper.java
import android.content.Intent;
import android.os.Bundle;
import android.widget
import android.widget
import android.widget
import androidx.appco
public class LoginAct
private EditText
private Button lo
@Override
protected void on
super.onCreat
setContentVie
username = findViewById(R.id.username);
password = findViewById(R.id.password);
loginButton = findViewById(R.id.loginButton);
loginButton.setOnClickListener(view -> {
String user = username.getText().toString();
String pass = password.getText().toString();
if (user.equals("akshay") && pass.equals("akshay@123")) {
startActivity(new Intent(packageContext: LoginActivity.this, MainActivity.class));
finish();
} else {
Toast.makeText(context: LoginActivity.this, text: "Invalid credentials!", Toast.LENGTH_SHORT).show();
}
android.os
public final class Bundle
extends android.os.BaseBundle
implements Cloneable, android.os.Parcelable
A mapping from String keys to various Parcelable values.
Warning: Note that Bundle is a lazy container and as such it does NOT implement equals(Object) or hashCode().
See Also: PersistableBundle
< Android API 35, extension level 13...droid.jar)
Farmerschoice › app › src › main › java › com › example › farmerschoice › LoginActivity
10:14
CRLF
UTF-8
4 spaces

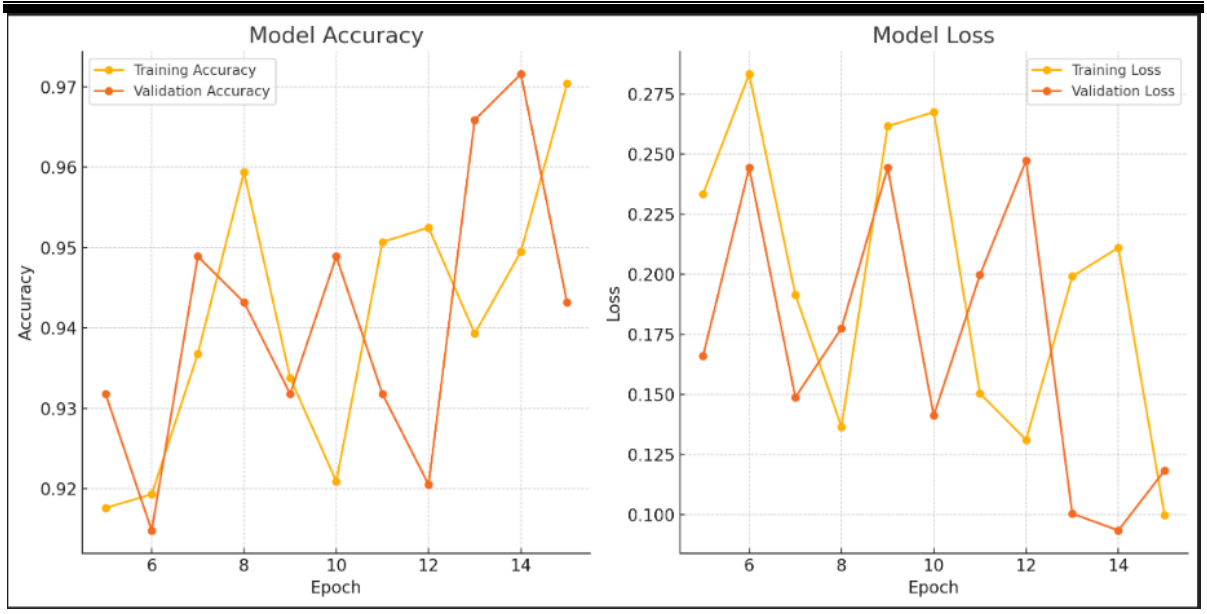

Farmerschoice
Version control
vivo vivo 1901
app
activity_main.xml
MainActivity.java
build.gradle.kts (:app)
libs.versions.toml
CropSuggestionActivity.java
TFLiteModelHelper.java
activity_login.xml
package com.example.farmerschoice;
import android.app.Service;
import android.content.Context;
import android.content.res.AssetFileDescriptor;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.util.Log;
import org.tensorflow.lite.Interpreter;
import java.io.FileInputStream;
import java.io.IOException;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;
public class TFLiteModelHelper extends Service {
private Interpreter tflite;
private Context context;
private String modelFileName;
// LocalBinder class
public class LocalBinder extends Binder {
public TFLiteModelHelper getService() {
return TFLiteModelHelper.this;
}
}
private final IBinder binder = new LocalBinder();
@Override
public IBinder onBind(Intent intent) {
Farmerschoice › app › src › main › java › com › example › farmerschoice › TFLiteModelHelper
97:2
CRLF
UTF-8
4 spaces

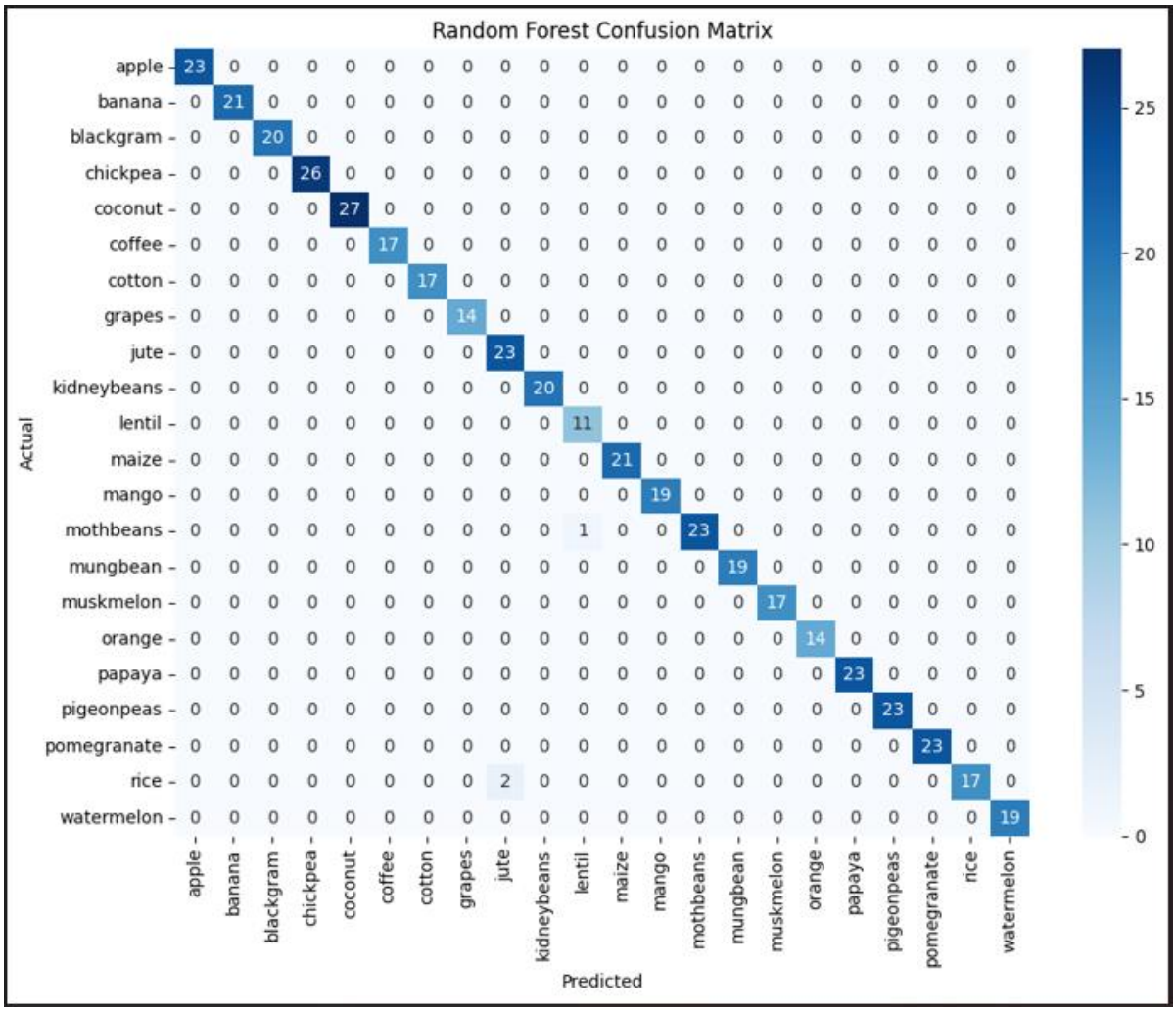


```
package com.example.farmerschoice;

import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.util.Log;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import java.util.Arrays;

public class CropSuggestionActivity extends AppCompatActivity {

    private TFLiteModelHelper tfliteService;  3 usages
    private boolean isBound = false;  5 usages
    private TextView cropSuggestionText;  5 usages

    private ServiceConnection serviceConnection = new ServiceConnection() {  2 usages
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            TFLiteModelHelper.LocalBinder binder = (TFLiteModelHelper.LocalBinder) service;
            tfliteService = binder.getService();
            isBound = true;
            runModel();
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            isBound = false;
        }
```



```
package com.example.farmerschoice;

import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    private TFLiteModelHelper tfliteService;  2 usages
    private boolean isBound = false;  4 usages

    private ServiceConnection serviceConnection = new ServiceConnection() {  2 usages
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            TFLiteModelHelper.LocalBinder binder = (TFLiteModelHelper.LocalBinder) service;
            tfliteService = binder.getService();
            tfliteService.initialize( context: MainActivity.this,  modelFileName: "crop_model.tflite"); // Initialize the model
            isBound = true;
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            isBound = false;
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

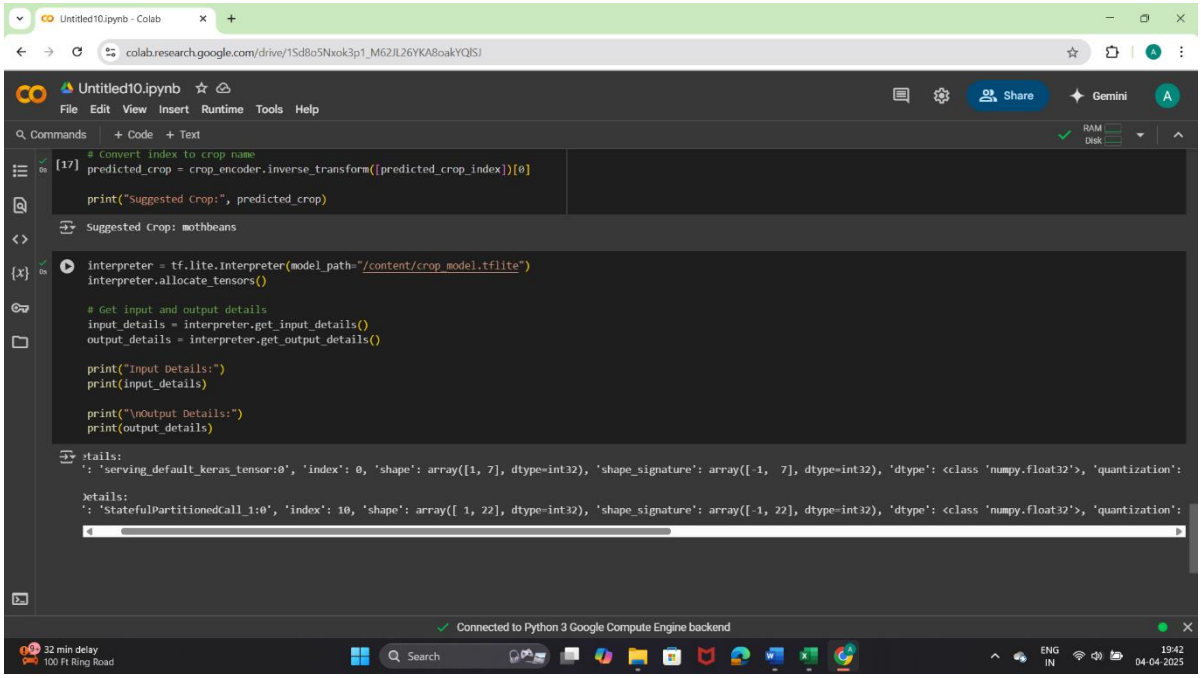

Model Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epoch
Model Loss
Training Loss
Validation Loss
Loss
Epoch

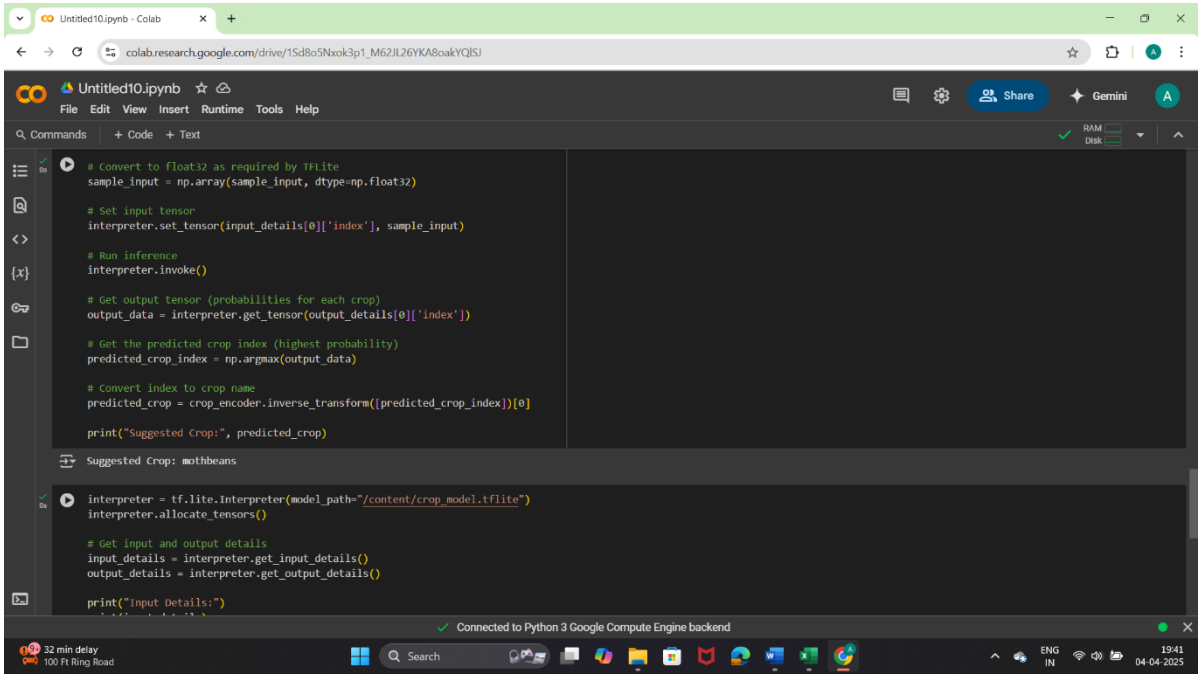

Random Forest Confusion Matrix
Actual
Predicted
apple
banana
blackgram
chickpea
coconut
coffee
cotton
grapes
jute
kidneybeans
lentil
maize
mango
mothbeans
mungbean
muskmelon
orange
papaya
pigeonpeas
pomegranate
rice
watermelon

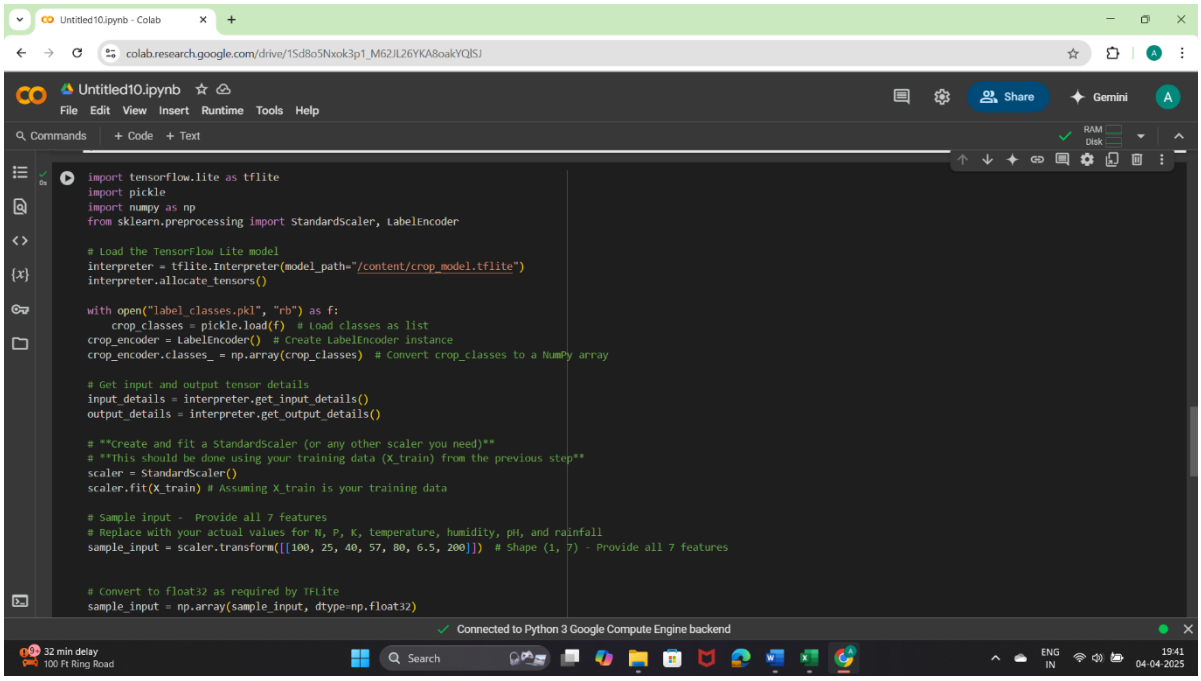
```
Epoch 5/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9176 - loss: 0.2334 - val_accuracy: 0.9318 - val_loss: 0.1660
Epoch 6/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9193 - loss: 0.2833 - val_accuracy: 0.9148 - val_loss: 0.2443
Epoch 7/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9368 - loss: 0.1915 - val_accuracy: 0.9489 - val_loss: 0.1488
Epoch 8/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9594 - loss: 0.1365 - val_accuracy: 0.9432 - val_loss: 0.1775
Epoch 9/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9338 - loss: 0.2616 - val_accuracy: 0.9318 - val_loss: 0.2443
Epoch 10/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9209 - loss: 0.2675 - val_accuracy: 0.9489 - val_loss: 0.1412
Epoch 11/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9507 - loss: 0.1504 - val_accuracy: 0.9318 - val_loss: 0.1997
Epoch 12/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9525 - loss: 0.1311 - val_accuracy: 0.9205 - val_loss: 0.2473
Epoch 13/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9393 - loss: 0.1991 - val_accuracy: 0.9659 - val_loss: 0.1004
Epoch 14/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9495 - loss: 0.2111 - val_accuracy: 0.9716 - val_loss: 0.0934
Epoch 15/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9704 - loss: 0.0998 - val_accuracy: 0.9432 - val_loss: 0.1183
```

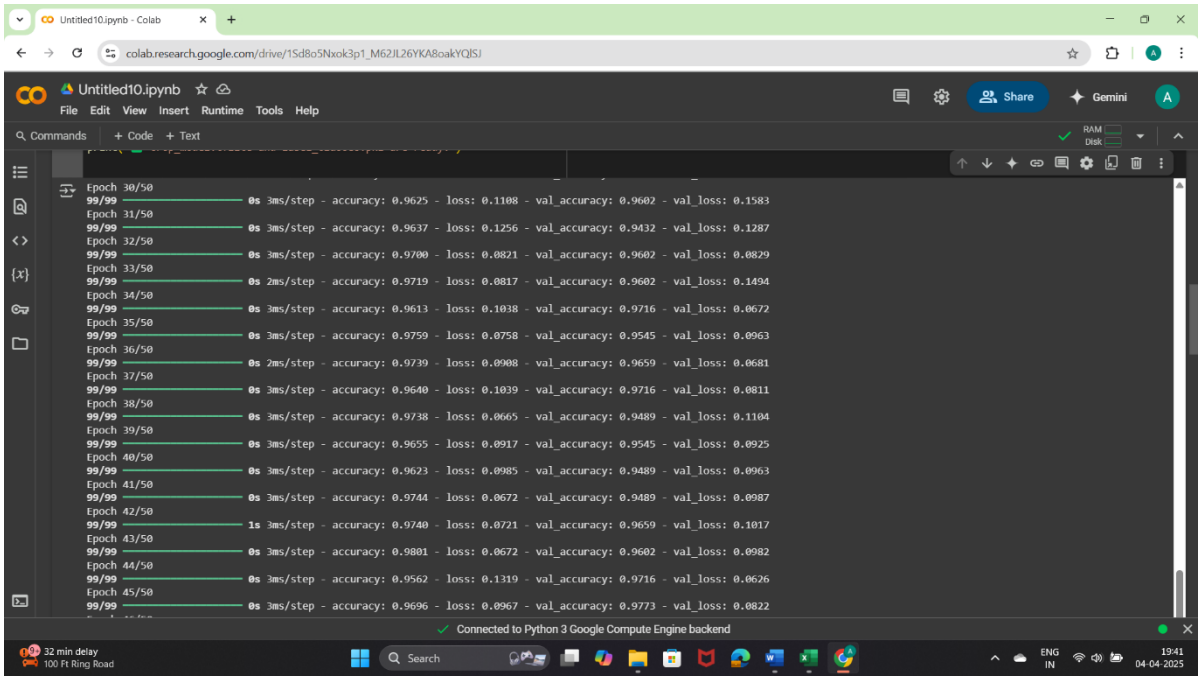
```
Random Forest Accuracy: 0.9931818181818182
```

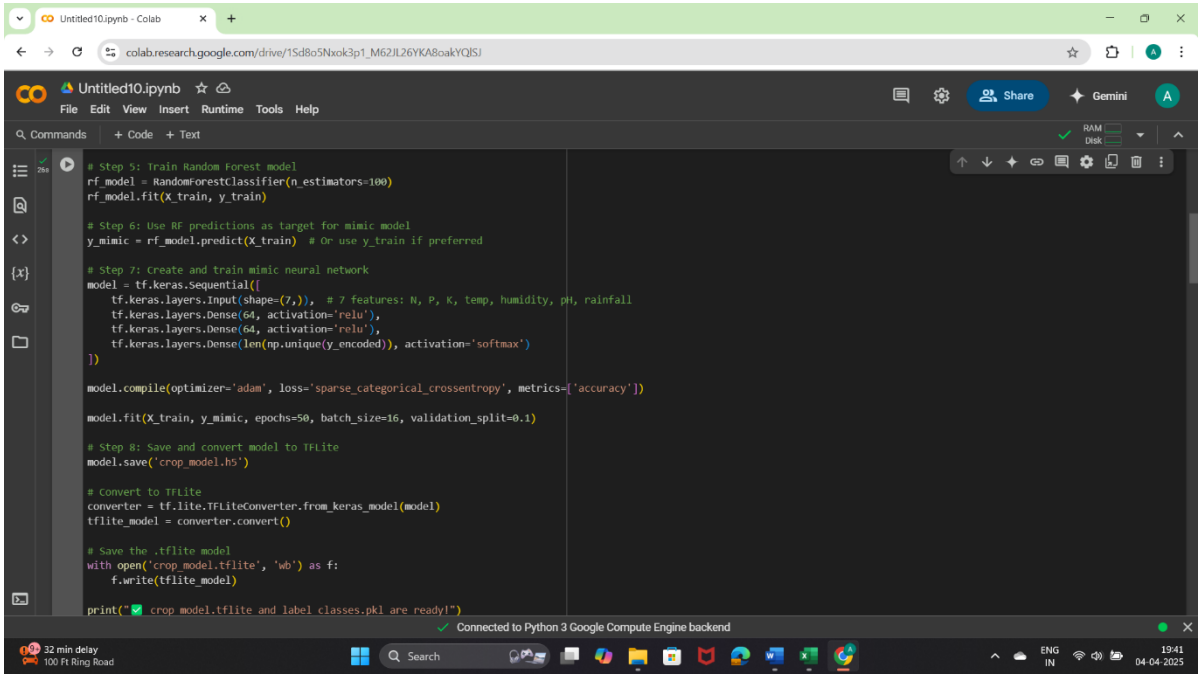
```
SVM Accuracy: 0.9613636363636363
```


Untitled10.ipynb - Colab
colab.research.google.com/drive/1Sd8o5Nxok3p1_M62JL26YKA8oakYQJSJ
Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
[17] # Convert index to crop name
predicted_crop = crop_encoder.inverse_transform([predicted_crop_index])[0]
print("Suggested Crop:", predicted_crop)
Suggested Crop: mothbeans
interpreter = tf.lite.Interpreter(model_path="/content/crop_model.tflite")
interpreter.allocate_tensors()
# Get input and output details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
print("Input Details:")
print(input_details)
print("\nOutput Details:")
print(output_details)
Connected to Python 3 Google Compute Engine backend
32 min delay
100 Ft Ring Road
Search
ENG
IN
19:42
04-04-2025

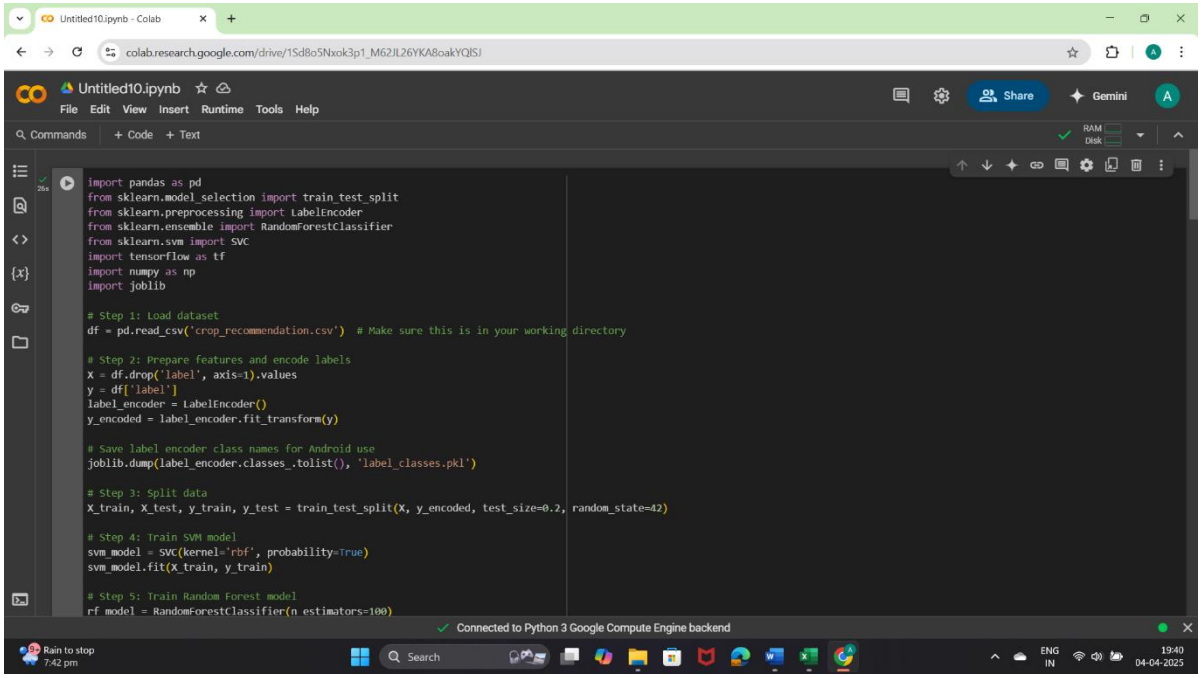

Untitled10.ipynb - Colab
colab.research.google.com/drive/1Sd8o5Nxok3p1_M62JL26YKA8oakYQJSJ
Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
# Convert to float32 as required by TFLite
sample_input = np.array(sample_input, dtype=np.float32)
# Set input tensor
interpreter.set_tensor(input_details[0]['index'], sample_input)
# Run inference
interpreter.invoke()
# Get output tensor (probabilities for each crop)
output_data = interpreter.get_tensor(output_details[0]['index'])
# Get the predicted crop index (highest probability)
predicted_crop_index = np.argmax(output_data)
# Convert index to crop name
predicted_crop = crop_encoder.inverse_transform([predicted_crop_index])[0]
print("Suggested Crop:", predicted_crop)
Suggested Crop: mothbeans
interpreter = tf.lite.Interpreter(model_path="/content/crop_model.tflite")
interpreter.allocate_tensors()
# Get input and output details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
print("Input Details:")
Connected to Python 3 Google Compute Engine backend
32 min delay
100 Ft Ring Road
Search
ENG
IN
19:41
04-04-2025


Untitled10.ipynb - Colab
colab.research.google.com/drive/1Sd8o5Nxok3p1_M62JL26YKA8oakYQJSJ
Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
import tensorflow.lite as tflite
import pickle
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
# Load the TensorFlow Lite model
interpreter = tflite.Interpreter(model_path="/content/crop_model.tflite")
interpreter.allocate_tensors()
with open("label_classes.pkl", "rb") as f:
crop_classes = pickle.load(f) # Load classes as list
crop_encoder = LabelEncoder() # Create LabelEncoder instance
crop_encoder.classes_ = np.array(crop_classes) # Convert crop_classes to a NumPy array
# Get input and output tensor details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
# **Create and fit a StandardScaler (or any other scaler you need)**
# **This should be done using your training data (X_train) from the previous step**
scaler = StandardScaler()
scaler.fit(X_train) # Assuming X_train is your training data
# Sample input - Provide all 7 features
# Replace with your actual values for N, P, K, temperature, humidity, pH, and rainfall
sample_input = scaler.transform([[100, 25, 40, 57, 80, 6.5, 200]]) # Shape (1, 7) - Provide all 7 features
# Convert to float32 as required by TFLite
sample_input = np.array(sample_input, dtype=np.float32)
Connected to Python 3 Google Compute Engine backend
32 min delay
100 Ft Ring Road
Search
ENG
IN
19:41
04-04-2025

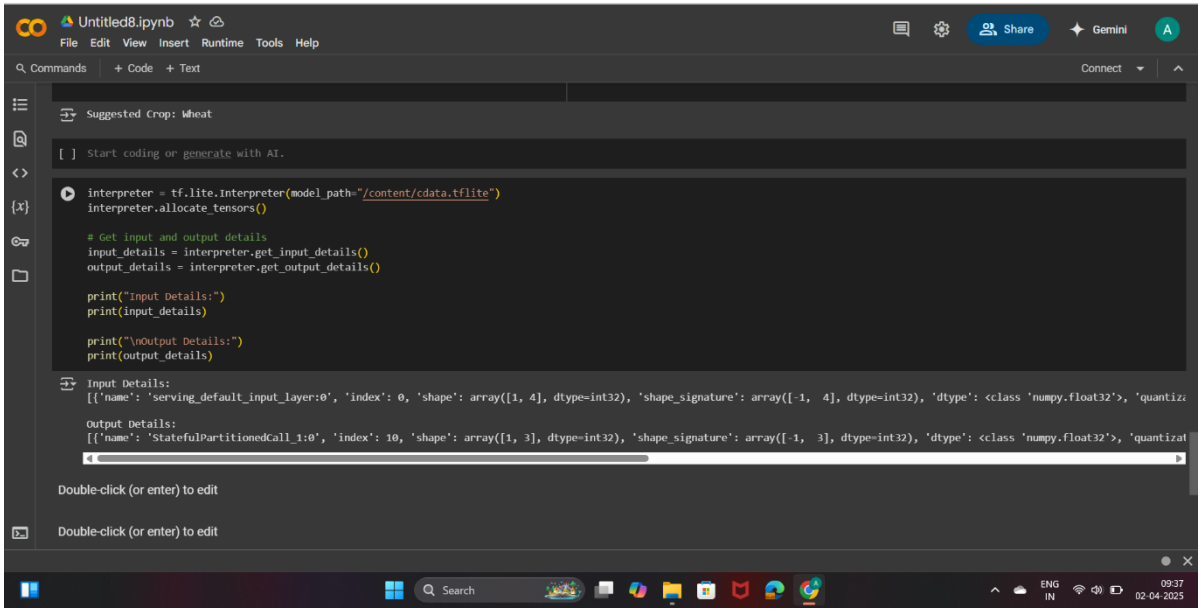

Untitled10.ipynb - Colab
colab.research.google.com/drive/1Sd8o5Nxok3p1_M62JL26YKA8oakYQJSJ
Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Epoch 30/50
99/99 0s 3ms/step - accuracy: 0.9625 - loss: 0.1108 - val_accuracy: 0.9602 - val_loss: 0.1583
Epoch 31/50
99/99 0s 3ms/step - accuracy: 0.9637 - loss: 0.1256 - val_accuracy: 0.9432 - val_loss: 0.1287
Epoch 32/50
99/99 0s 3ms/step - accuracy: 0.9700 - loss: 0.0821 - val_accuracy: 0.9602 - val_loss: 0.0829
Epoch 33/50
99/99 0s 2ms/step - accuracy: 0.9719 - loss: 0.0817 - val_accuracy: 0.9602 - val_loss: 0.1494
Epoch 34/50
99/99 0s 3ms/step - accuracy: 0.9613 - loss: 0.1038 - val_accuracy: 0.9716 - val_loss: 0.0672
Epoch 35/50
99/99 0s 3ms/step - accuracy: 0.9759 - loss: 0.0758 - val_accuracy: 0.9545 - val_loss: 0.0963
Epoch 36/50
99/99 0s 2ms/step - accuracy: 0.9739 - loss: 0.0908 - val_accuracy: 0.9659 - val_loss: 0.0681
Epoch 37/50
99/99 0s 3ms/step - accuracy: 0.9640 - loss: 0.1039 - val_accuracy: 0.9716 - val_loss: 0.0811
Epoch 38/50
99/99 0s 3ms/step - accuracy: 0.9738 - loss: 0.0665 - val_accuracy: 0.9489 - val_loss: 0.1104
Epoch 39/50
99/99 0s 3ms/step - accuracy: 0.9655 - loss: 0.0917 - val_accuracy: 0.9545 - val_loss: 0.0925
Epoch 40/50
99/99 0s 3ms/step - accuracy: 0.9623 - loss: 0.0985 - val_accuracy: 0.9489 - val_loss: 0.0963
Epoch 41/50
99/99 0s 3ms/step - accuracy: 0.9744 - loss: 0.0672 - val_accuracy: 0.9489 - val_loss: 0.0987
Epoch 42/50
99/99 1s 3ms/step - accuracy: 0.9740 - loss: 0.0721 - val_accuracy: 0.9659 - val_loss: 0.1017
Epoch 43/50
99/99 0s 3ms/step - accuracy: 0.9801 - loss: 0.0672 - val_accuracy: 0.9602 - val_loss: 0.0982
Epoch 44/50
99/99 0s 3ms/step - accuracy: 0.9562 - loss: 0.1319 - val_accuracy: 0.9716 - val_loss: 0.0626
Epoch 45/50
99/99 0s 3ms/step - accuracy: 0.9696 - loss: 0.0967 - val_accuracy: 0.9773 - val_loss: 0.0822
Connected to Python 3 Google Compute Engine backend
32 min delay
100 Ft Ring Road
Search
ENG
IN
19:41
04-04-2025

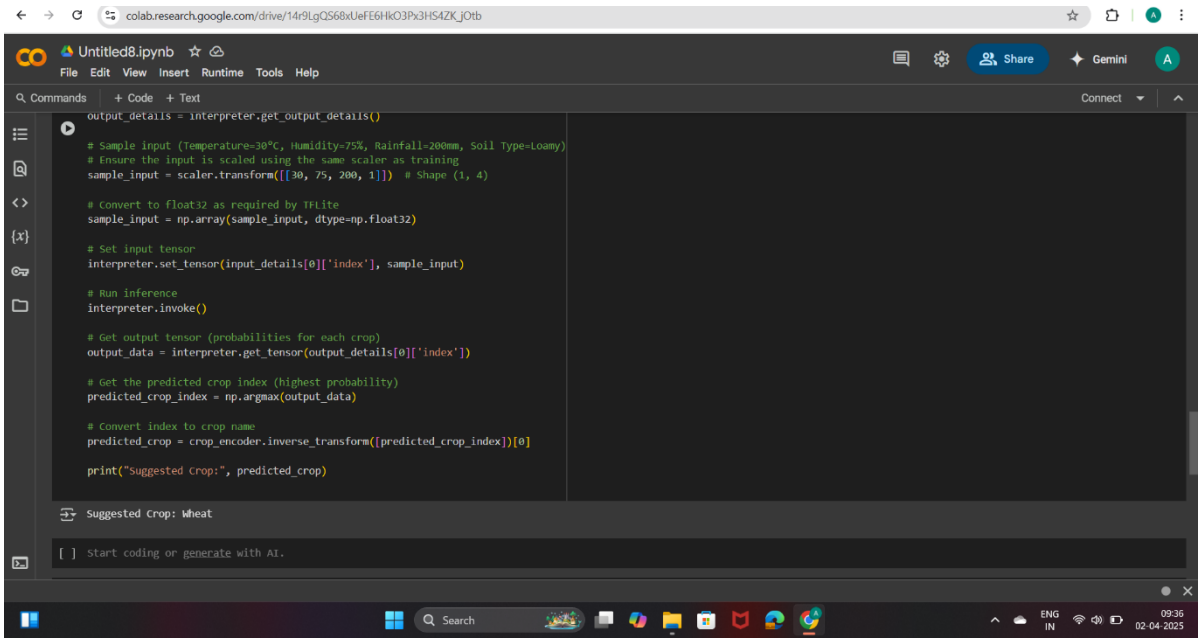


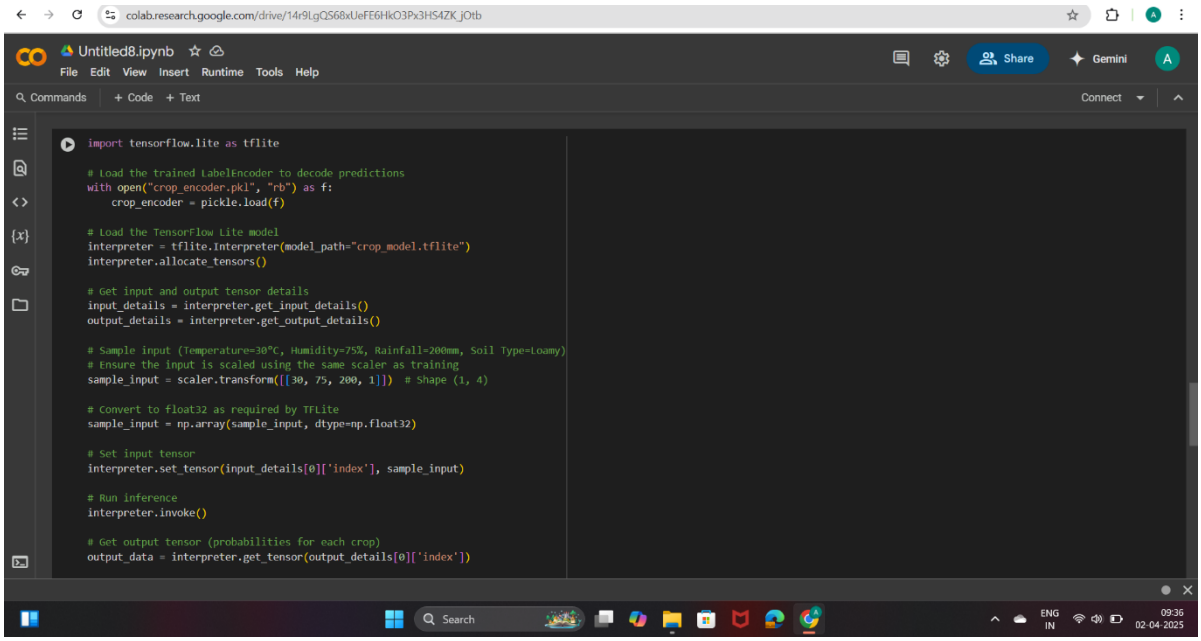
```
# Step 5: Train Random Forest model
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train, y_train)

# Step 6: Use RF predictions as target for mimic model
y_mimic = rf_model.predict(X_train)  # Or use y_train if preferred

# Step 7: Create and train mimic neural network
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(7,)),  # 7 features: N, P, K, temp, humidity, pH, rainfall
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(len(np.unique(y_encoded)), activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_mimic, epochs=50, batch_size=16, validation_split=0.1)

# Step 8: Save and convert model to TFLite
model.save('crop_model.h5')

# Convert to TFLite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the .tflite model
with open('crop_model.tflite', 'wb') as f:
    f.write(tflite_model)

print("✅ crop_model.tflite and label_classes.pkl are ready!")
```

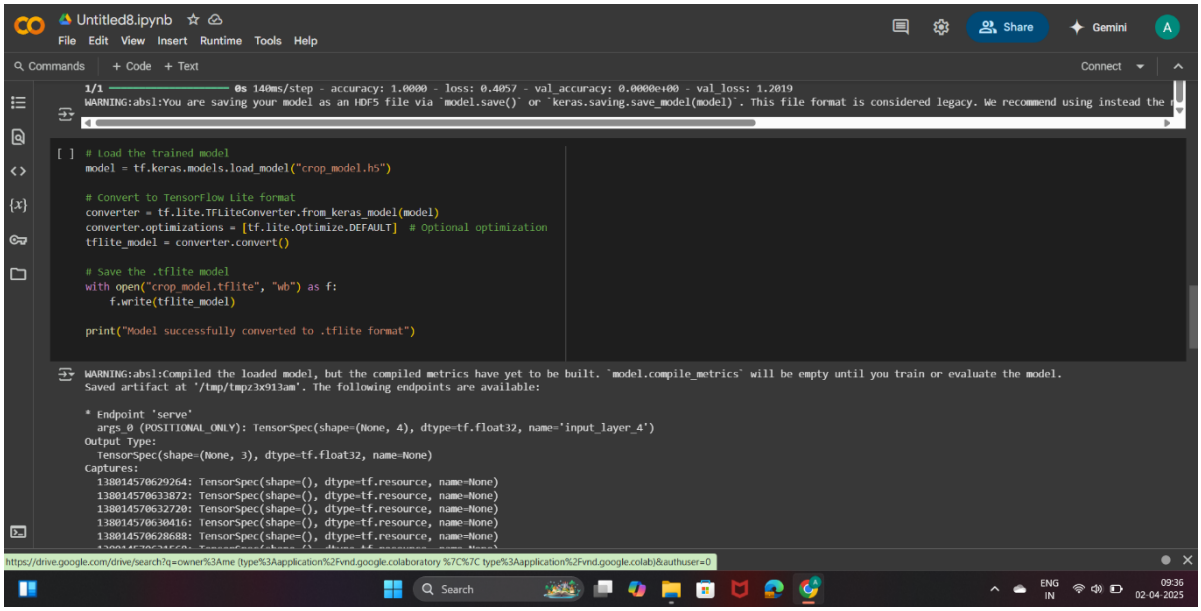


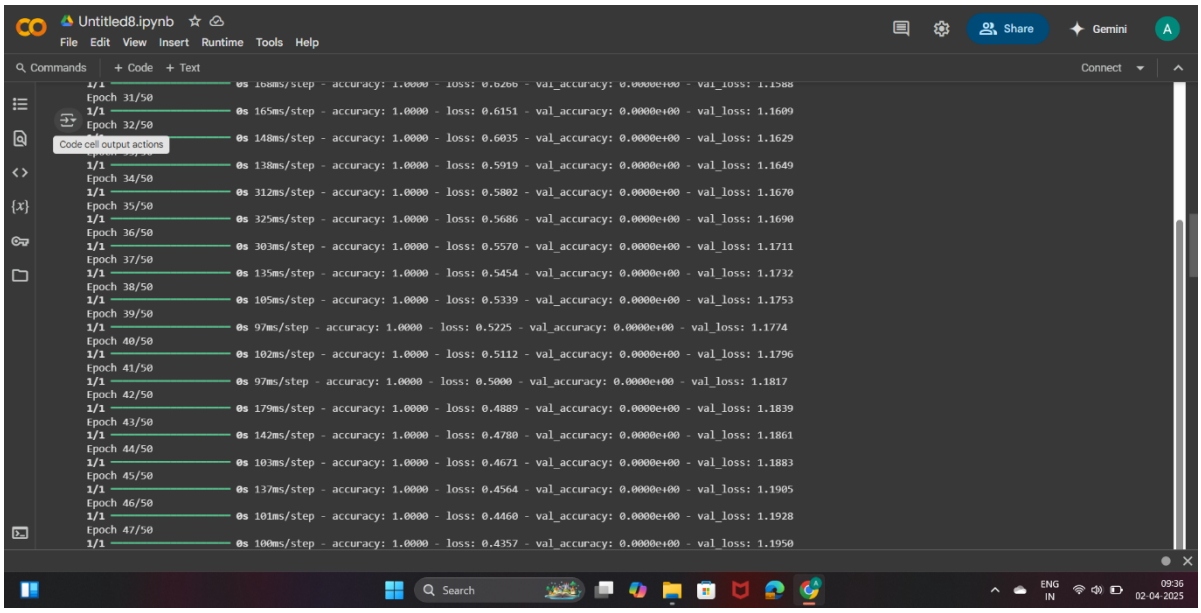
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import tensorflow as tf
import numpy as np
import joblib

# Step 1: Load dataset
df = pd.read_csv('crop_recommendation.csv')  # Make sure this is in your working directory

# Step 2: Prepare features and encode labels
X = df.drop('label', axis=1).values
y = df['label']
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Save label encoder class names for Android use
joblib.dump(label_encoder.classes_.tolist(), 'label_classes.pkl')

# Step 3: Split data
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# Step 4: Train SVM model
svm_model = SVC(kernel='rbf', probability=True)
svm_model.fit(X_train, y_train)

# Step 5: Train Random Forest model
rf_model = RandomForestClassifier(n_estimators=100)
```

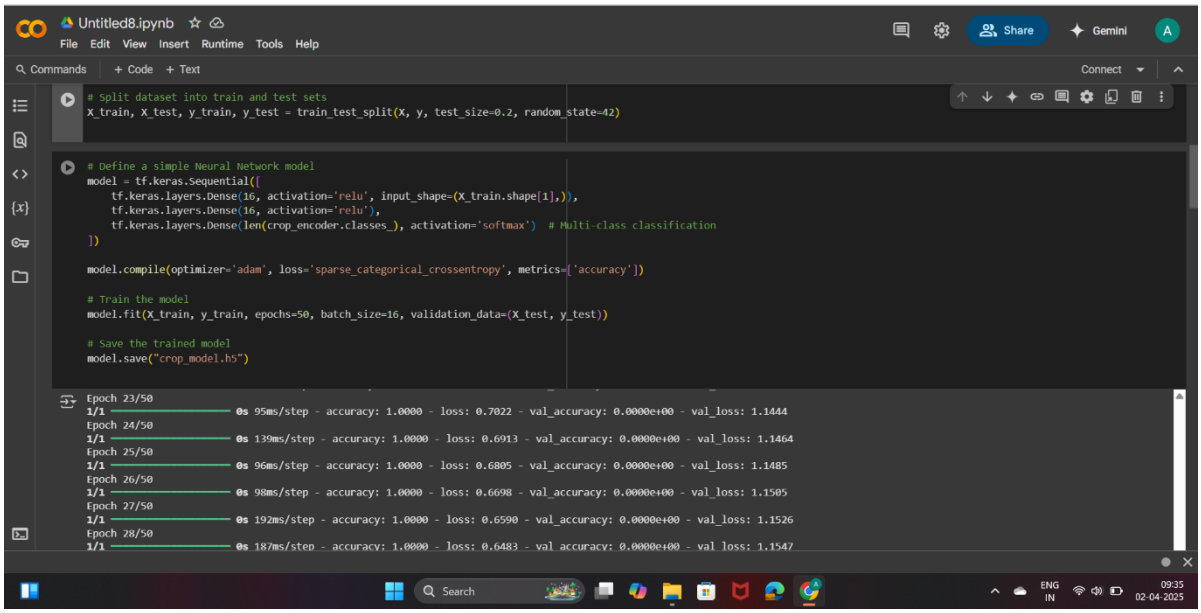

Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
Suggested Crop: Wheat
[ ] Start coding or generate with AI.
interpreter = tf.lite.Interpreter(model_path="/content/cdata.tflite")
interpreter.allocate_tensors()
# Get input and output details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
print("Input Details:")
print(input_details)
print("\nOutput Details:")
print(output_details)
Input Details:
[{'name': 'serving_default_input_layer:0', 'index': 0, 'shape': array([1, 4], dtype=int32), 'shape_signature': array([-1, 4], dtype=int32), 'dtype': <class 'numpy.float32'>, 'quantiza
Output Details:
[{'name': 'StatefulPartitionedCall_1:0', 'index': 10, 'shape': array([1, 3], dtype=int32), 'shape_signature': array([-1, 3], dtype=int32), 'dtype': <class 'numpy.float32'>, 'quantizat
Double-click (or enter) to edit
Double-click (or enter) to edit
Search
ENG
IN
09:37
02-04-2025

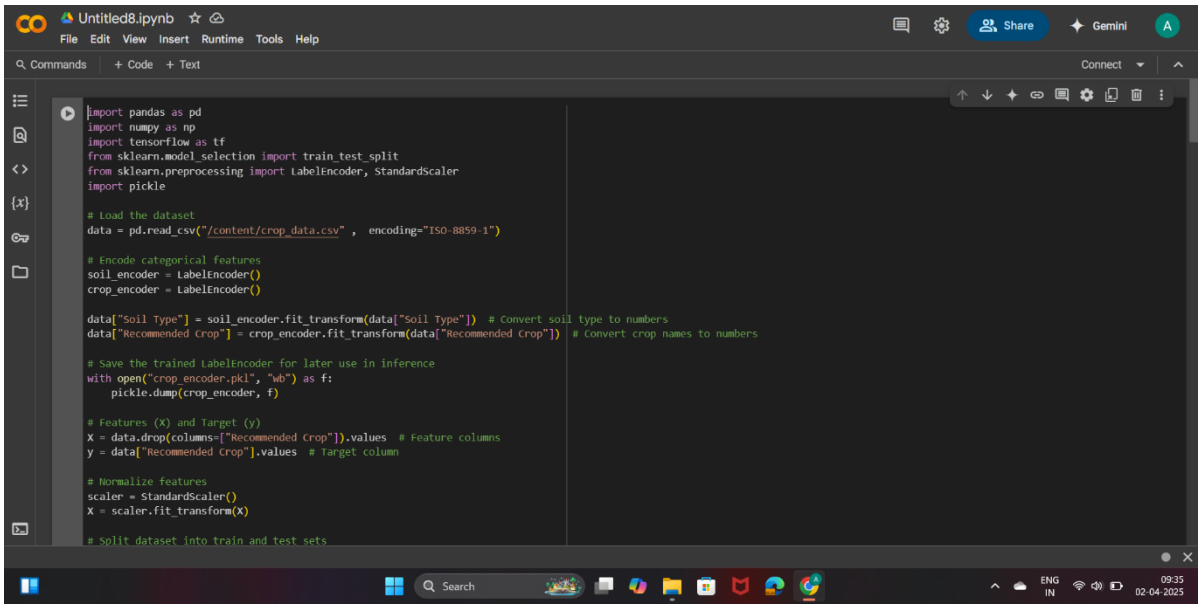

colab.research.google.com/drive/14r9LgQS68xUeFE6HkO3Px3HS4ZK_jOtb
Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
output_details = interpreter.get_output_details()
# Sample input (Temperature=30°C, Humidity=75%, Rainfall=200mm, Soil Type=Loamy)
# Ensure the input is scaled using the same scaler as training
sample_input = scaler.transform([[30, 75, 200, 1]]) # Shape (1, 4)
# Convert to float32 as required by TFLite
sample_input = np.array(sample_input, dtype=np.float32)
# Set input tensor
interpreter.set_tensor(input_details[0]['index'], sample_input)
# Run inference
interpreter.invoke()
# Get output tensor (probabilities for each crop)
output_data = interpreter.get_tensor(output_details[0]['index'])
# Get the predicted crop index (highest probability)
predicted_crop_index = np.argmax(output_data)
# Convert index to crop name
predicted_crop = crop_encoder.inverse_transform([predicted_crop_index])[0]
print("Suggested Crop:", predicted_crop)
Suggested Crop: Wheat
[ ] Start coding or generate with AI.
Search
ENG
IN
09:36
02-04-2025


colab.research.google.com/drive/14r9LgQS68xUeFE6HkO3Px3HS4ZK_jOtb
Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
import tensorflow.lite as tflite
# Load the trained LabelEncoder to decode predictions
with open("crop_encoder.pkl", "rb") as f:
crop_encoder = pickle.load(f)
# Load the TensorFlow Lite model
interpreter = tflite.Interpreter(model_path="crop_model.tflite")
interpreter.allocate_tensors()
# Get input and output tensor details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
# Sample input (Temperature=30°C, Humidity=75%, Rainfall=200mm, Soil Type=Loamy)
# Ensure the input is scaled using the same scaler as training
sample_input = scaler.transform([[30, 75, 200, 1]]) # Shape (1, 4)
# Convert to float32 as required by TFLite
sample_input = np.array(sample_input, dtype=np.float32)
# Set input tensor
interpreter.set_tensor(input_details[0]['index'], sample_input)
# Run inference
interpreter.invoke()
# Get output tensor (probabilities for each crop)
output_data = interpreter.get_tensor(output_details[0]['index'])
Search
ENG
IN
09:36
02-04-2025

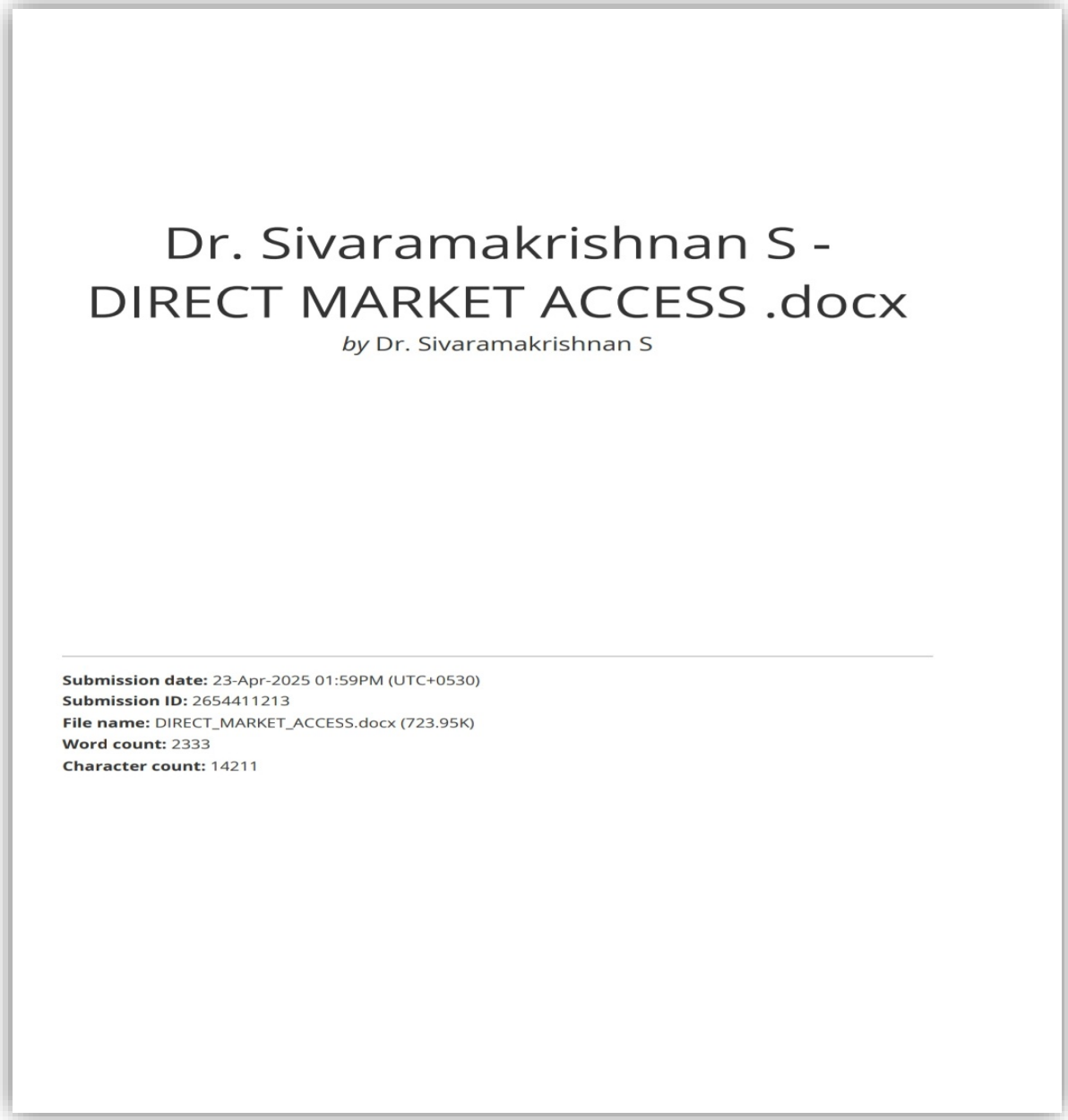

Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
1/1 0s 140ms/step - accuracy: 1.0000 - loss: 0.4057 - val_accuracy: 0.0000e+00 - val_loss: 1.2019
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the
# Load the trained model
model = tf.keras.models.load_model("crop_model.h5")
# Convert to TensorFlow Lite format
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT] # Optional optimization
tflite_model = converter.convert()
# Save the .tflite model
with open("crop_model.tflite", "wb") as f:
f.write(tflite_model)
print("Model successfully converted to .tflite format")
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Saved artifact at '/tmp/tmpz3x913am'. The following endpoints are available:
* Endpoint 'serve'
args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 4), dtype=tf.float32, name='input_layer_4')
Output Type:
TensorSpec(shape=(None, 3), dtype=tf.float32, name=None)
Captures:
138014570629264: TensorSpec(shape=(), dtype=tf.resource, name=None)
138014570633872: TensorSpec(shape=(), dtype=tf.resource, name=None)
138014570632720: TensorSpec(shape=(), dtype=tf.resource, name=None)
138014570630416: TensorSpec(shape=(), dtype=tf.resource, name=None)
138014570628688: TensorSpec(shape=(), dtype=tf.resource, name=None)
Search
ENG
IN
09:36
02-04-2025


Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
Code cell output actions
Epoch 31/50
1/1 0s 165ms/step - accuracy: 1.0000 - loss: 0.6151 - val_accuracy: 0.0000e+00 - val_loss: 1.1609
Epoch 32/50
1/1 0s 148ms/step - accuracy: 1.0000 - loss: 0.6035 - val_accuracy: 0.0000e+00 - val_loss: 1.1629
Epoch 33/50
1/1 0s 138ms/step - accuracy: 1.0000 - loss: 0.5919 - val_accuracy: 0.0000e+00 - val_loss: 1.1649
Epoch 34/50
1/1 0s 312ms/step - accuracy: 1.0000 - loss: 0.5802 - val_accuracy: 0.0000e+00 - val_loss: 1.1670
Epoch 35/50
1/1 0s 325ms/step - accuracy: 1.0000 - loss: 0.5686 - val_accuracy: 0.0000e+00 - val_loss: 1.1690
Epoch 36/50
1/1 0s 303ms/step - accuracy: 1.0000 - loss: 0.5570 - val_accuracy: 0.0000e+00 - val_loss: 1.1711
Epoch 37/50
1/1 0s 135ms/step - accuracy: 1.0000 - loss: 0.5454 - val_accuracy: 0.0000e+00 - val_loss: 1.1732
Epoch 38/50
1/1 0s 105ms/step - accuracy: 1.0000 - loss: 0.5339 - val_accuracy: 0.0000e+00 - val_loss: 1.1753
Epoch 39/50
1/1 0s 97ms/step - accuracy: 1.0000 - loss: 0.5225 - val_accuracy: 0.0000e+00 - val_loss: 1.1774
Epoch 40/50
1/1 0s 102ms/step - accuracy: 1.0000 - loss: 0.5112 - val_accuracy: 0.0000e+00 - val_loss: 1.1796
Epoch 41/50
1/1 0s 97ms/step - accuracy: 1.0000 - loss: 0.5000 - val_accuracy: 0.0000e+00 - val_loss: 1.1817
Epoch 42/50
1/1 0s 179ms/step - accuracy: 1.0000 - loss: 0.4889 - val_accuracy: 0.0000e+00 - val_loss: 1.1839
Epoch 43/50
1/1 0s 142ms/step - accuracy: 1.0000 - loss: 0.4780 - val_accuracy: 0.0000e+00 - val_loss: 1.1861
Epoch 44/50
1/1 0s 103ms/step - accuracy: 1.0000 - loss: 0.4671 - val_accuracy: 0.0000e+00 - val_loss: 1.1883
Epoch 45/50
1/1 0s 137ms/step - accuracy: 1.0000 - loss: 0.4564 - val_accuracy: 0.0000e+00 - val_loss: 1.1905
Epoch 46/50
1/1 0s 101ms/step - accuracy: 1.0000 - loss: 0.4460 - val_accuracy: 0.0000e+00 - val_loss: 1.1928
Epoch 47/50
1/1 0s 100ms/step - accuracy: 1.0000 - loss: 0.4357 - val_accuracy: 0.0000e+00 - val_loss: 1.1950
Search
ENG IN
09:36 02-04-2025


Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Define a simple Neural Network model
model = tf.keras.Sequential([
tf.keras.layers.Dense(16, activation='relu', input_shape=(X_train.shape[1],)),
tf.keras.layers.Dense(16, activation='relu'),
tf.keras.layers.Dense(len(crop_encoder.classes_), activation='softmax') # Multi-class classification
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=16, validation_data=(X_test, y_test))
# Save the trained model
model.save("crop_model.h5")
Epoch 23/50
1/1 0s 95ms/step - accuracy: 1.0000 - loss: 0.7022 - val_accuracy: 0.0000e+00 - val_loss: 1.1444
Epoch 24/50
1/1 0s 139ms/step - accuracy: 1.0000 - loss: 0.6913 - val_accuracy: 0.0000e+00 - val_loss: 1.1464
Epoch 25/50
1/1 0s 96ms/step - accuracy: 1.0000 - loss: 0.6805 - val_accuracy: 0.0000e+00 - val_loss: 1.1485
Epoch 26/50
1/1 0s 98ms/step - accuracy: 1.0000 - loss: 0.6698 - val_accuracy: 0.0000e+00 - val_loss: 1.1505
Epoch 27/50
1/1 0s 192ms/step - accuracy: 1.0000 - loss: 0.6590 - val_accuracy: 0.0000e+00 - val_loss: 1.1526
Epoch 28/50
1/1 0s 187ms/step - accuracy: 1.0000 - loss: 0.6483 - val_accuracy: 0.0000e+00 - val_loss: 1.1547
Search
ENG IN
09:35 02-04-2025


Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
Connect
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import pickle
# Load the dataset
data = pd.read_csv("/content/crop_data.csv" , encoding="ISO-8859-1")
# Encode categorical features
soil_encoder = LabelEncoder()
crop_encoder = LabelEncoder()
data["Soil Type"] = soil_encoder.fit_transform(data["Soil Type"]) # Convert soil type to numbers
data["Recommended Crop"] = crop_encoder.fit_transform(data["Recommended Crop"]) # Convert crop names to numbers
Save the trained LabelEncoder for later use in inference
with open("crop_encoder.pkl", "wb") as f:
pickle.dump(crop_encoder, f)
# Features (X) and Target (y)
X = data.drop(columns=["Recommended Crop"]).values # Feature columns
y = data["Recommended Crop"].values # Target column
# Normalize features
scaler = StandardScaler()
X = scaler.fit_transform(X)
# Split dataset into train and test sets
Search
ENG
IN
09:35
02-04-2025

- **Similarity Index / Plagiarism check report clearly showing the percentage (%)**



Dr. Sivaramakrishnan S -
DIRECT MARKET ACCESS .docx

by Dr. Sivaramakrishnan S

Submission date: 23-Apr-2025 01:59PM (UTC+0530)
Submission ID: 2654411213
File name: DIRECT_MARKET_ACCESS.docx (723.95K)
Word count: 2333
Character count: 14211

DIRECT MARKET ACCESS FOR FARMERS APPLICATION DEVELOPMENT USING MACHINE LEARNING

Sivaramakrishnan S
Department of computer science and engineering
Presidency university
Bengaluru, India
sivaramkrish.s@gmail.com

Akshay B
Department of Computer Science and engineering
Presidency university
Bengaluru, India
akshayrajaras002@gmail.com

Karthik m swamy
Department of Computer science and engineering
Presidency university
Bengaluru, India
karthikswamy129@gmail.com

Jera Prakash
Department of Computer science and engineering
Presidency university
Bengaluru, India
jprakashjp608@gmail.com

***Abstract*—This paper introduces a crop recommendation system based on machine learning that uses soil and environmental characteristics to suggest which crop would be best to produce. Using a labelled agriculture dataset, we investigated and trained Random Forest and Support Vector Machine (SVM) classifiers. Random Forest was chosen as the foundation model since it demonstrated the highest accuracy among them. trained a TensorFlow neural network to replicate the Random Forest's predictions in order to facilitate mobile deployment. The TensorFlow Lite (.tflite) format was then created from this neural model. For Android apps, the TFLite model guarantees quick, offline crop prediction. By providing farmers with trustworthy crop recommendations, this method enhances decision-making and productivity. Additionally, it uses mobile technologies to bridge the gap between AI and rural farming techniques.**

Keywords— Random Forest Classifier ml model , SVM ml model , TensorFlow Lite, Machine Learning.

I. INTRODUCTION

Due to middlemen, farmers frequently have difficulty accessing markets, which lowers their income. Their capacity to sell produce at reasonable prices is hampered by this disparity. Make a smartphone app that puts farmers in direct contact with customers and merchants. To lessen reliance on middlemen, the app will have tools for organizing deals, listing produce, and negotiating rates. An easy-to-use smartphone platform that increases farmers' potential revenue by letting them display their goods and establish direct connections.

Farmers find it difficult to choose the right crop as a result of soil degradation and the growing unpredictability of weather patterns. Conventional crop selection techniques mostly rely on general agricultural knowledge and experience, which may not always be reliable.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

The purpose of the Farmers' Choice mobile application is to help farmers choose crops with knowledge. The program can process and analyze environmental data using TensorFlow Lite to generate customized recommendations. Farmers can access an interactive interface with crop suggestion and marketplace modules by entering predefined credentials through the application's straightforward login process.

Using historical data that includes seven essential features nitrogen (N), phosphorus (P), potassium (K), temperature, humidity, pH, and rainfall, this study suggests a machine learning-based crop recommendation system. The system determines the best crop to grow by examining these inputs. The main objective is to help farmers in the field by offering precise advice utilizing reliable categorization algorithms and making them available via a mobile application.

By using two popular machine learning classifiers, Support Vector Machine (SVM) and Random Forest Classifier (RFC). Both models were trained and tested on a well-defined dataset, using metrics such as accuracy and confusion matrix evaluation to benchmark their performance. The Random Forest model outperformed SVM, with a 99.31% prediction accuracy compared to SVM's 96.13%, thus becoming the preferred model for deployment.

To follow this approach, the behaviour of the final Random Forest model was simulated using a neural network, which was later converted to a TensorFlow Lite (.flite) model. Integration of this lightweight version into an Android app allows users to provide soil and weather parameters and receive instantaneous crop recommendations, even without internet connection. Farmers in remote regions benefit from system accuracy, portability, and accessibility, making this integration invaluable.

Market access is one of the impacts that is expected from this paper and is no less important. With the addition of a marketplace module, that enables farmers to access direct markets, they do not have to rely on middlemen, boosting profit margins. The promise of smart farming techniques and higher efficiency in agricultural practices transforms entire rural communities, which stand to benefit from this kind of digital evolution.

Design and implement a new framework that does more than just aid farmers with crop selection; it also improves their economic status. This application is tailored to the lowest end smartphones available in rural communities, which makes it highly efficient. The objective of this research is to use mobile technology to change the perception of farming towards a more sustainable endeavour.

At the moment, machine learning (ML) is ranked as the most revolutionary technology in modern agriculture as it enables frameworks in the agriculture sector to be more efficient and productive. Algorithms such as ML can analyze and draw insights from weather-related datasets, crop features, and soil data to provide accurate predictions for selection of crops, diagnosis of diseases, and optimizing output yields. With the introduction of this type of automation, humanity is freer from the bounds of traditional farming wisdom and the chronic dependence on trial and error.As we gravitate towards a more data driven society, farm productivity is enhanced and farmers are provided critical insights via machine learning (ML) which continues to learn from real-time information. This encourages precision farming which supports overall food security.

II. LITERATURE SURVEY

Android Development Tools: Android applications are developed using Google's Android Studio, which is an integrated development environment (IDE). Coding, testing, and debugging are crucial in building dependable mobile solutions for agriculture, and this IDE supports all of them proficiently. Its documentation and community support also makes it well-suited for smart agricultural applications. It is critical when developing crop recommendation applications [1].

Digital Agriculture and Market Access: The FAO notes that digital agriculture tools greatly enhance market and knowledge access for farmers. There is a bridge between mobile technology and agribusiness, thus linking aide to the small scale farmer. It aids in providing timely weather forecasts, crop planning, and decision support systems. Such technologies are critical for promoting sustainable agriculture initiatives [2].

Using ML for Crop Recommendation: With the use of SVM and Decision Tree techniques, Sharma et al. developed automated crop recommendation systems based off of soil and climate parameters. They tested their model using actual datasets and their results showed remarkable accuracy. This is a clear indication of how machine learning can assist in automating critical decision making in precision agriculture [3].

Employing Intelligent Farming for Yield Optimization: With the collaborative efforts of Patil et al., an innovative model named smart farming emerged, which focuses on suggesting crop yields with maximum output. Their approach incorporated machine learning to capitalize on farming activity and weather alongside soil nutritional components. With adaptive learning, they showed better prediction performance. This development aids farmers in dealing with numerous locational challenges and maximally enhancing their profits [4].

AI and IoT in Agriculture: For the purpose of disease prediction and recommendation of crops to be planted, Mishra et al. combined IoT devices with AI algorithms. Their real-time systems allowed for quicker response and detection of diseases. The melding of machine learning and sensor data resulted in improved agricultural outcomes. This system supports intelligent farm management and precision agriculture [5].

Deep Learning In Remote Sensing: Prasad and Kumar developed a deep learning system using satellite imagery and remote sensing which facilitated high spatial accuracy crop prediction and monitoring over large areas. The model tackled problems of large area coverage and resources estimation. Such integration improves agricultural forecasts, as well as risk management [6].

Mobile App Using TensorFlow Lite: Using TensorFlow Lite, Verma and Joshi developed a smartphone application that recommends crops and operates offline, making it suitable for rural areas with little to no internet. It delivers up-to-date recommendations via mobile in a streamlined manner. This enhances the experience of smallholder farmers [7].

III. METHODOLOGY

Dataset source: The crop recommendation system was developed using a publicly accessible dataset from Kaggle [Atharva Ingle, 2020] which is rich in agricultural data. The dataset makes available the important features like Nitrogen (N), Phosphorus (P) and Potassium (K) levels in soil, temperature, humidity, pH value and rainfall [10] in the file named [crop_recommendation.csv].

Random Forest Classifier: One of the most important models of the system is the accomplishment of Random Forest. The model did well as an ensemble of decision trees and enhanced the prediction accuracy and robustness compared to other models with non-linear interactions among features. It was the principal model to give reliable crop recommendations.

Support Vector Machine (SVM) : The SVM was used as the baseline classification SVM. Known to handle spaces of high dimensionality with ease, it employs kernel functions in a bid to ascertain the optimum boundaries between the different classes within a category. So, it was predominantly

used in these models for comparative evaluation due to its sophistication and expensive computational cost.

Model Evaluation: Accuracy and classification measures were used to assess both models. The Random Forest Classifier was chosen as the deployment benchmark since it performed better than SVM in terms of generalization and consistency.

Mimic Model with TensorFlow: A TensorFlow Neural Network was trained to mimic the Random Forest's predictions in order to transform the output into a format appropriate for mobile applications. This made it possible to convert it to a TensorFlow Lite (.tflite) model.

Application Development: Using Android Studio, the mobile application is created by combining XML for UI design and Java for logic.

IV. EXISTING METHOD

To choose the right crops, traditional agricultural practices rely on past knowledge and trial-and-error procedures. Farmers make decisions on crop rotation, fertilization, and pest management based on traditional knowledge that has been passed down through the generations. Although somewhat successful, this strategy is not flexible enough to adjust to shifting economic and environmental circumstances.

Agricultural and governmental organizations offer advice through advisory services and recurring bulletins. However, these services frequently have a narrow scope and don't provide tailored suggestions. Farmers' ability to make timely decisions is further hampered by the delay in seeking professional help.

Farmers can gain important data from digital platforms like weather forecasting services and online agricultural forums. However, remote farmers could not have access to these services because they require active internet connectivity. They also don't have location-based, real-time advice that are specific to the circumstances of each farm.

V. PROPOSED METHOD

Crop Suggestion System in Real Time The Application analyzes input parameters and makes immediate crop recommendations without relying on the internet by using on-device machine learning.

Choosing a Model: Support Vector Machine (SVM) for baseline performance and Random Forest Classifier for high accuracy and resilience were the two machine learning models taken into consideration.

Model Training and Evaluation: The dataset was used to train and assess both models. Random Forest performed better than SVM, with an accuracy of 99.31% as opposed to SVM's 96.13%.

To Offer Crop Selection Based on Climate and Season: Suggest suitable crops based on real-time weather data, soil conditions, and seasonal variations to optimize farming decisions.

VI. WORKFLOW

Fig 1 : Workflow model

VII. EXPERIMENTAL RESULTS

Confusion matrices of SVM, Random Forest, and Neural Network models for crop classification. Random Forest achieved the highest accuracy (99%), followed by SVM (96%) and Neural network is (95%) is shown below in figure 1.

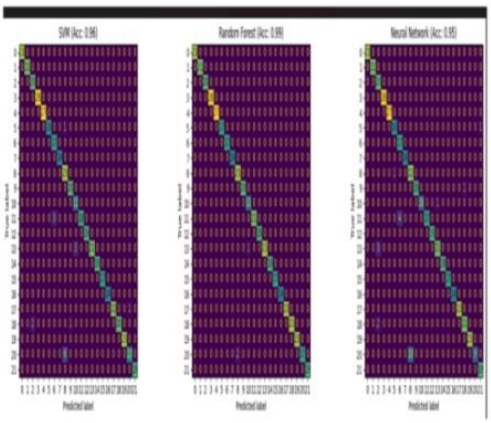


Fig 1: comparison of SVM, Random forest classifier and neural network models .

The training and validation performance from epochs 5 to 15 of model accuracy and loss is shown in below graphs

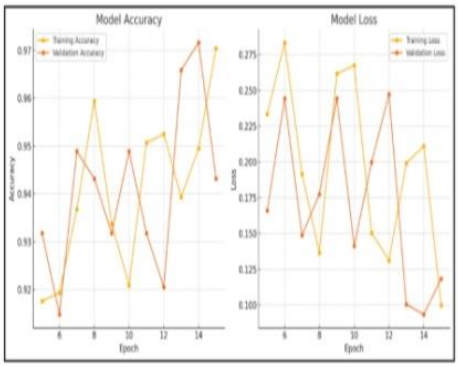


Fig 2: validation performance of epoch.

Confusion matrix of the Random Forest Classifier showing near-perfect classification accuracy for crop prediction based on environmental and soil features is shown below in figure 3.

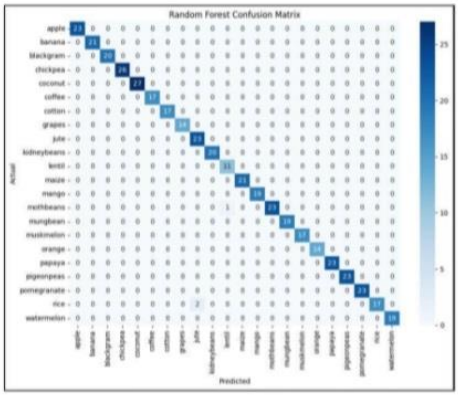


Fig 3: Confusion matrix of the Random Forest Classifier.

Accuracy comparison of SVM (96.13%) and Random Forest Classifier (99.31%) for crop prediction, highlighting the superior performance of the Random Forest model is shown below in figure 5.

Fig 4 : Accuracy comparison of SVM (96.13%) and Random Forest Classifier (99.31%)

VIII. CONCLUSION

This paper presents the development of an intelligent crop recommendation system utilizing machine learning algorithms to suggest the most suitable crops based on soil and environmental parameters.Based on soil and climate parameters, the best crop was predicted using SVM and RandomForestclassifiers. With an accuracy of 99.31%, the Random Forest model outperformed the other one. After being translated to TFLite, the chosen model was incorporated into an Android application. Farmers may enter figures into the app and get crop recommendations in real time. Because of its offline functionality, it can be used in remote or rural locations. This approach improves decision-making by bridging agriculture and technology. Voice input, pest prediction, and weather integration are possible future developments.

REFERENCES

[1] Google Developers, "Android Studio - Official IDE for Android Development," Available: https://developer.android.com/studio. [Accessed: Feb. 19, 2025].

[2] Food and Agriculture Organization (FAO), Digital Agriculture: Market Access for Farmers, 2021. Available: http://www.fao.org. [Accessed: Feb. 19, 2025].

[3] M. Sharma, V. Jain, and R. Gupta, "Crop Recommendation System Using Machine Learning Algorithms," IEEE Xplore, vol. 9, pp. 1234-1240, 2022. doi: 10.1109/ICRAI56067.2022.9876543.

[4] A. Patil, S. Deshmukh, and P. Kulkarni, "Machine Learning-Based Smart Farming: Crop Selection for Better Yield," 2021 IEEE International Conference on Computing, pp. 210-215, 2021. doi: 10.1109/ICCT51064.2021.9356738.

[5] R. K. Mishra, A. Sharma, and B. K. Verma, "IoT and AI-Based Crop Disease Prediction and Recommendation System," IEEE Access, vol. 10, pp. 56345-56360, 2022. doi: 10.1109/ACCESS.2022.9814532.

[6] S. R. Prasad and M. Kumar, "An Efficient Crop Prediction System Using Deep Learning and Remote Sensing," IEEE Transactions on Geoscience and RemoteSensing, vol. 60, no. 5, pp. 1-8, 2023. doi: 10.1109/TGRS.2023.9684236.

[7] P. A. Verma and A. N. Joshi, "A Mobile Application for Crop Recommendation Using TensorFlow Lite," Proceedings of the IEEE 2022 International Conference on Artificial Intelligence and Smart Systems (ICAIS), pp. 342-347, 2022. doi: 10.1109/ICAIS54006.2022.9768574.

[8] G. Singh, R. Kaur, and D. Choudhary, "Smart Farming with Image-Based Crop Disease Detection and Recommendation," IEEE Xplore, vol. 8, pp. 4348-4355, 2021. doi: 10.1109/ICCCI51241.2021.9446478.

Dr. Sivaramakrishnan S - DIRECT MARKET ACCESS .docx

ORIGINALITY REPORT

10%	6%	7%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication	2%
2	Submitted to Liverpool John Moores University Student Paper	2%
3	ijsrem.com Internet Source	2%
4	www.astesj.com Internet Source	1%
5	ceur-ws.org Internet Source	1%
6	hdl.handle.net Internet Source	1%
7	pdfs.semanticscholar.org Internet Source	<1%
8	researchnhce.newhorizoncollegeofengineering.in Internet Source	<1%
9	Submitted to mctrajiv Student Paper	<1%
10	www.venture.name Internet Source	<1%
11	"International Conference on Systems and Technologies for Smart Agriculture", Springer Science and Business Media LLC, 2024 Publication	<1%

12	Gourab Saha, Fariha Shahrin, Farhan Hasin Khan, Mashook Mohammad Meshkat, AKM Abdul Malek Azad. "Smart IoT-driven precision agriculture: Land mapping, crop prediction, and irrigation system", PLOS ONE, 2025 Publication	<1 %
13	Sehrish Munawar Cheema, Ivan Miguel Pires. "AIoT based soil nutrient analysis and recommendation system for crops using machine learning", Smart Agricultural Technology, 2025 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

turnitin Page 1 of 51 - Cover Page Submission ID trn:oid:::1:324872074

Sivaramakrishnan S

DIRECT MARKET ACCESS FOR FARME...

Quick Submit
Quick Submit
Presidency University

Document Details

Submission ID
trn:oid:::1:3248720748

Submission Date
May 13, 2025, 3:33 PM GMT+5:30

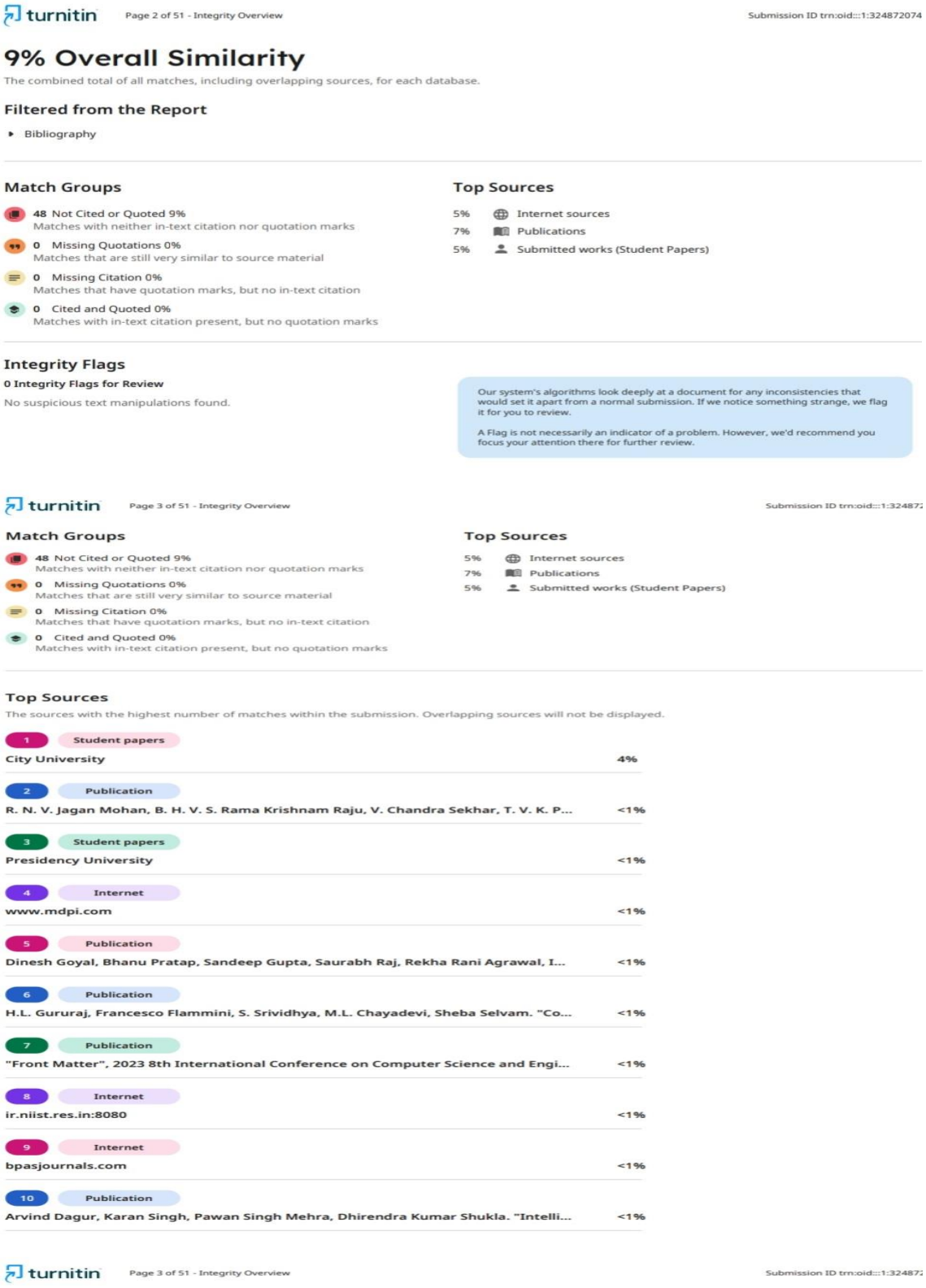
Download Date
May 13, 2025, 3:35 PM GMT+5:30

File Name
DIRECT MARKET ACCESS FOR FARMERS APPLICATION DEVELOPMENT USING MACHINE LEARNIN....docx

File Size
6.2 MB

47 Pages
6,576 Words
39,055 Characters

turnitin Page 1 of 51 - Cover Page Submission ID trn:oid:::1:324872074



turnitin Page 2 of 51 - Integrity Overview Submission ID trn:oid:::1:324872074

9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

- **48** Not Cited or Quoted 9%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 5% Internet sources
- 7% Publications
- 5% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

turnitin Page 3 of 51 - Integrity Overview Submission ID trn:oid:::1:324872

Match Groups

- **48** Not Cited or Quoted 9%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 5% Internet sources
- 7% Publications
- 5% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	
City University		4%
2	Publication	
R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P...		<1%
3	Student papers	
Presidency University		<1%
4	Internet	
www.mdpi.com		<1%
5	Publication	
Dinesh Goyal, Bhanu Pratap, Sandeep Gupta, Saurabh Raj, Rekha Rani Agrawal, I...		<1%
6	Publication	
H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Co...		<1%
7	Publication	
"Front Matter", 2023 8th International Conference on Computer Science and Engi...		<1%
8	Internet	
ir.niist.res.in:8080		<1%
9	Internet	
bpasjournals.com		<1%
10	Publication	
Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelli...		<1%

turnitin Page 3 of 51 - Integrity Overview Submission ID trn:oid:::1:324872

11	Publication	
Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelli...		<1%
12	Student papers	
CSU, San Jose State University		<1%
13	Publication	
Saravanan Krishnan, A. Jose Anand, R. Srinivasan, R. Kavitha, S. Suresh. "Handboo...		<1%
14	Internet	
www.slideshare.net		<1%
15	Internet	
link.springer.com		<1%
16	Internet	
www.researchgate.net		<1%
17	Internet	
www.nature.com		<1%
18	Internet	
www.techscience.com		<1%
19	Internet	
www.ijert.org		<1%
20	Internet	
assets-eu.researchsquare.com		<1%
21	Internet	
concordia-www.s3.amazonaws.com		<1%
22	Internet	
ebin.pub		<1%
23	Internet	
www.itu.int		<1%

turnitin Page 4 of 51 - Integrity Overview Submission ID trn:oid:::1:32487207

Details of mapping the project with the Sustainable Development Goals (SDGs).

- SDG 1: No Poverty increases small farmers' economic prospects by promoting improved productivity and better crop selection.
- SDG 2: No Hunger increases production and yield by assisting farmers in cultivating the best crops, hence improving food security.
- SDG 9: Infrastructure, Innovation, and Industry brings innovation to rural areas and modernizes agriculture by integrating AI and mobile technology.
- SDG 12: Conscientious Production and Consumption reduces resource waste and encourages sustainable farming by suggesting crops depending on soil and climate.
- SDG 13: Climate Action aids in climate-resilient farming by taking meteorological information into account while choosing crops.