Analyzing Socio Economic Development Using Deep Learning on Satellite Imagery

A Project Report

Presented to

The Faculty of the Department of Applied Data Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Data Analytics

By

Ahaz Bhatti

Deep Arvind Bambharoliya

Shrey Bishnoi

Syama Ravi Teja Jerrypothula

May 19, 2023

APPROVED FOR DEPARTMENT OF APPLIED DATA SCIENCE

Dr. Eduardo Chan, Project Advisor

Dr. Lee C. Chang, Department Chair

ABSTRACT

Analyzing Socio Economic Development Using Deep Learning on Satellite Imagery

By

Ahaz Bhatti, Deep Arvind Bambharoliya, Shrey Bishnoi, Syama Ravi Teja Jerrypothula

Purpose

Global poverty levels have significantly increased in recent years and continue to rise. 689 million people, or 9.2% of the world's population, live in extreme poverty, according to a 2021 study. The daily income of these people is less than $1.90. While there is a solution to this problem, its impact is limited because most countries lack reliable data required to implement effective policies. There are various methods to collect accurate information on asset determination. It began with the collection of personal data and has now evolved into large-scale data gathering using smart devices such as satellites, computers, and smartphones. Traditional research is expensive, labor-intensive, and time-consuming. Therefore, more efficient techniques are needed to collect this data, along with other metrics like income rate and employment rate. Satellite images can also provide insights into the socioeconomic status of a location. This raises the question of why we cannot assess wealth using visual data. By combining satellite images taken during the day and nighttime, we can create a highly accurate model for evaluating wealth in relation to poverty. Such a model would make it easier for regulatory bodies to obtain labor statistics, saving time, money, and providing a more effective method. It would enable better tracking of wealth, aiding in the development of plans and policies to strengthen the economy.

Tasks

We are implementing the CRISP-DM in this project. The project is divided into 6 key stages using this manner. The first step is to understand the problem better by gathering all the

research information on this topic. Then the next step would be to gather trustworthy data. We require two types of data for this project, namely satellite images and economic data on local poverty levels. With the help of Google static maps, high resolution satellite imagery data will be gathered and survey data from the LSMS or DHS websites. These images take up many gigabytes, thus processing power is essential to execute this project. We plan to leverage Amazon Web Services to satisfy these needs. Understanding the data and moving on to the next phases, needs a lot of time once it has been collected. The following step would be to use Amazon SageMaker to pre-process this data. In this stage, we will first clean the data using pre-processing techniques and then segment the images using Python libraries. After that, we will combine the survey data and the segmented images. In order to create a deep learning model, we will retrieve the processed data in SageMaker after the data has been saved in the S3 bucket. The models will then be assessed using evaluation metrics, and the best model will be deployed on a web portal so that we can learn from it. This project's whole information, including visualizations comparing model performances, will be available on this web portal. Furthermore, we aim to inculcate a feature where the user can upload an image to the web portal to gain poverty status of a particular location considering the parameters used to train a model.

Outcomes

After predicting the level of development in many different countries of Africa by implementing deep learning on satellite imagery data, an online platform will be deployed that provides a brief overview of the data collection and analysis processes used at the national level. Additionally, it will have a feature which illustrates day and night satellite imagery showing the extent of poverty in each country. The data will be described on the web portal along with the deep learning models that were used to analyze it and sections that provide all of the

visualizations. Additionally, a code implementation containing the whole functional protocol to execute these models, evaluate them, and validate them will be supplied. Furthermore, a final report and presentation will be made, providing a brief summary of the subject and the analysis we will conduct throughout the duration of the project. The goal is to create a model that uses high resolution satellite imagery and deep learning to forecast the socioeconomic state of an area so that the government may concentrate on the areas that may require development. Additionally, this will offer uncommon opportunities for long-term economic indicator collection in developing countries, which frequently necessitates costly field research. Different hyperparameters and fine tuning will be utilized to compare the various deep learning models that will be employed. Metrics including accuracy, precision, and AUROC curve will be used to assess each model's correctness. We will use survey data from the LSMS (Living Standards Measurement Study) or DHS (Demographic and Health Surveys) to validate our models and assess how well they forecast the amount of poverty in a country, as they may be more accurate than field surveys.

Applications

In this research, we use deep learning algorithms to examine and detect the living circumstances of a country. The web portal that will be created will display the statistics and results for various locations. Government agencies, as well as other for-profit groups, will be able to concentrate on the regions where assistance is needed with the use of this online portal. Additionally, it enables the organizations to better focus social welfare and humanitarian relief, enlighten disadvantaged groups, and assess the results of policy actions. Additionally, it enables various trusts, non-governmental organizations, and philanthropists to focus on areas where they may offer humanitarian aid. It can also help countries in reducing the cost of surveys performed

for analyzing the development by effectively using the results based on satellite images which are available at much lower costs in the public domain.

ACKNOWLEDGMENTS

Thank you to everyone who helped us complete this project, we appreciate all the support and guidance provided to us.

A special thanks to Dr. Eduardo Chan. He did an amazing job guiding us and sharing his knowledge with us throughout the project. Dr. Chan's knowledge in data analytics played an important role in the completion of our project.

Additionally, we want to thank our department chairperson Dr. Lee Chang for his support. It has been an honor for him in monitoring the progress and completion of our project.

Our peers, friends, and family members helped us financially with this project and inspired us throughout this project. Their encouragement and belief in our abilities have been a motivating source of inspiration.

Lastly, we are grateful to all those who have contributed to the success of this project. Their support, guidance, and contributions have played a big role in its completion, and we are honored to have had the opportunity to work with such incredible individuals and organizations.

TABLE OF CONTENTS

Chapter 4 Model Development

4.1 Model Proposals

4.2 Model Supports

4.3 Model Comparison and Justification

4.4 Model Evaluation Methods

4.5 Model Validation and Evaluation Results

Chapter 5 Data Analytics and Intelligent System

5.1 System Requirements Analysis

5.2 System Design

5.3 Intelligent Solution

5.4 System Supporting Environment

Chapter 6 System Evaluation and Visualization

6.1 Analysis of Model Execution and Evaluation Results

6.2 Achievements and Constraints

6.3 System Quality Evaluation of Model Functions and Performance

6.4 System Visualization

Chapter 7 Conclusion

7.1 Summary

7.2 Benefits and Shortcoming

7.3 Potential System and Model Applications

Analyzing Socio Economic Development Using Deep Learning on Satellite Imagery

Project Background and Execute Summary

Poverty is defined as the inability to meet basic needs such as food, clothing and shelter. It is one of the important parameters that needs to be estimated when analyzing the economy of a region or country. Economic surveys have long been used to get an idea of the economic situation of local people using this information governing bodies can develop strategies and policies to improve living standards. Governing bodies have historically relied on investigations of all kinds which consume significant amounts of time, money and human resources. Hence, using this approach to determine budget allocations and other policy making processes is costly and time consuming as the issue has led researchers to look for alternative strategies to assess economic viability in resource poor populations as described above without sacrificing efficacy.

The primary goal is to calculate poverty levels using satellite imagery data as there are numerous aspects that can be taken into account when evaluating a location's economic situation basically there are differences between economically wealthy and poor regions in a number of parameters including financial outlays housing costs employment rates and revenue generated in many sectors including business entertainment malls and so forth. Everyone of the aforementioned variables is typically higher or better in economically wealthy regions than in less wealthy regions as it was previously indicated in relation to conventional surveys some of the parameters are challenging to gather. The quality and reliability of the data is also a consideration for the other metrics such as revenue generation from various sectors.

In the realm of information technology, artificial intelligence (AI) is one of the hottest topics. AI is about the idea of human imitation and behavior, with the aim of substituting a machine for a human to complete a task. Numerous jobs might be automated using this idea

without affecting performance, and in certain circumstances, it could even outperform earlier methods while using fewer resources. Machine learning is a branch of artificial intelligence that deals with the idea of automatically learning from data and creating models to guide decision-making. Human intervention in machine learning occurs to a certain extent, including feature selection and other things. Deep learning, on the other hand, is a different idea that also avoids that. Deep learning works with a broad range of data, including voice, pictures, and video. This type of data typically demands a powerful computer and a complicated algorithm. Deep learning can accomplish that. Our project uses satellite photos, so we planned to use deep learning to tackle this tough aspect of the project.

The governments themselves are the project's potential end users. By saving them time, money, and human resources, our project will enable them to estimate a location's richness. They will use those findings to inform new policies and/or initiatives they develop to better the situation where they are located. In addition, using a straightforward satellite image of the location will enable one to monitor the present state of wealth at a specific time. With this strategy, gathering data won't require much work, and the reliability and quality of the data will be unaffected. Our will become simple and effective with this step alone.

Project Requirements

Data

Our project requires data related to financial assessment of a place with all the parameters which may help to understand them better for efficient use. We need satellite images of those places to train our model. For better performance we need both daytime and nighttime images of those places. Nighttime images need to be in a format of tagged images with the coordinates of the places so that it will be easy for us to segment the images of required places. So, we need two

different kinds of datasets for our project as mentioned earlier. The first is information about poverty in that location, and the second is high resolution satellite images of various locations. Both sets of data must be used to train the created model. There are various ways for us to obtain photographs. The DMSP-OLS website is one of them (Ncei, n.d). This program gathers information on population and health in developing nations. Access to datasets is permitted with their consent. In addition to that, the Google Static Maps API is another well-liked source of data. Either of the datasets will be used in our research. The additional data, which includes information from surveys on poverty, can be gathered from a variety of sources, including Data World, the Census Bureau, the World Bank, and others.

Functional Requirements

Our project is about assessing the socio-economic status of a place using satellite images of that particular place. We need a web application with user friendly User Interface (UI) with all the conclusions of observations drawn from the data. In the same website, the developed best deep learning model will be deployed. This website should also have an option to upload an image of a place and to predict the poverty of that place using the deployed model. This website needs to be hosted on cloud services like AWS. The application needs to be secured using the login page and every action of the user needs to be saved in the database for accounting purposes. These all are the important functional requirements of our project.

Our AI system needs to have efficient data to train the model. In our case we need plenty of satellite images of a place to train the model with the wealth index. Apart from this data, we need a model to be deployed on the website which has the ability to predict the wealth index using the uploaded satellite image.

Technology

A suitable pipeline is required to carry out any project. The project's non-functional requirements also have a significant impact on how successfully it is carried out. In this project, we are dealing with image data, tabular data, data preprocessing, transformation, and building a model using deal learning. In order to share our thoughts and monitor the strategy and development of our project, we must first establish a suitable communication platform. JIRA is a project management application that we use for this, and we also use apps like Zoom and WhatsApp for communication. We need a strong platform to compute on because deep learning is what we are working with. To develop a pipeline for that, we are utilizing the Amazon AWS platform.

Domain Knowledge

This project uses economic concepts that emphasize the value of evaluating a location's poverty levels. Understanding the economy is a broad subject, which is difficult. The phrase "poverty" is the primary emphasis of our study. From location to location, the definition of poverty varies slightly. That depends on the way of life of the inhabitants of that location, such as the number of calories needed for survival there, which depends on many of their long-standing customs. Therefore, that determines the poverty line or definition for that area. Determining the lines of extreme poverty is another topic up for discussion. Since a majority of those living in poverty defined as having a daily income of less than $30 fall into this category. Those that make ten times as little money as $30 per day are thus unable to distinguish between the two. Consequently, we must gather all pertinent facts (Roser, 2021).

Project Deliverables

The project's goal is to develop a system for forecasting socioeconomic circumstances in various African nations, which will help policymakers better target social welfare and humanitarian aid, educate underprivileged people, and evaluate the effects of policy efforts. The system will display a map of the various economic circumstances across a nation across various time periods.

The socio-economic status in various areas for the selected country is predicted by a model, system link, which takes in ground-level survey information, satellite images containing data about a country's economic conditions, and images of night and day, respectively, as input. The model then provides a visualization based on the prediction with respect to the time chosen by the user. A thorough project report that includes the project's background, the technical needs, a literature study, a plan for managing the data and the project, models for the data process, a plan for developing the platform, and an evaluation and deployment strategy. a repository on Github including the produced model, code for data processing, and stored model, as well as thorough documentation like the project report, presentation, and project specifications. The project's whole dataset, which includes survey data from DMSP-OLS website (Ncei, n.d) and satellite images captured throughout the day and at night using Google Maps' static API, is also given. Slides for the project presentation that provide thorough justifications of every project step as well as a demonstration of every action taken to achieve the project results. The project's deliverables are listed in Table 1 along with the anticipated deployment dates.

Table 1

Project Deliverables With Descriptions And Dates

No.	Phase	Deliverables	Description	Date
1.	Project Understanding	Literature Survey Report	This contains the description and learnings of the survey papers related to our problem and the conclusions and summary for each	03-October-2022
		Technology and data requirements report	Will evaluate the kind of technology and data needed for this project.	08-October 2022
		Project Plan	It includes the the development methodology including milestones, requirements and data management plan	14-October 2022
2.	Data Preparation	Data collection and EDA	gathering necessary information with numerous criteria from the necessary sources as well as in depth data exploration	15-October 2022

No.	Phase	Deliverables	Description	Date
		Data Modeling report	Preprocessing of data and finalizing the dataset for feeding to the deep learning models	15-September 2022
3.	Documentation	Project Presentation	Presenting the progress of project for the fall semester	07-December 2022
		Fall Semester report	A detailed project report about the progress made until the fall semester	11-December 2022
4.	Modeling	Model proposal and development report	Proposed models for the problem along with architecture, data flow, and technologies used for development	13-January 2023
		Models accuracy and evaluation report	Model accuracy and evaluation report using the various selected performance metrics	18-March 2023
5.	Evaluation	Comparison Report	Comparing the developed models and reporting the results of the evaluation metrics	28-March 2023

No.	Phase	Deliverables	Description	Date
		Summary report	Summarizing the performances and research	30-April 2023
6.	Deployment	Prototype Web application	A web portal which will host the model and present an output based on the user input	02-April 2023
		Presentation	Microsoft Powerpoint will be used to create the presentation.	10-May 2023 (Tentative)
		Final Report	A report for the whole project	12-May-2023 (Tentative)

Note. Project deliverables with the estimated date.

Technology and Solution Survey

In socio economic evaluations of a region or a nation, poverty is one of the key parameters that must be estimated. Macroeconomic studies have been employed for a long time to determine the livelihood of the native population. Many governments have relied primarily on surveys which use resources, including cost, time, and manpower. In order to decide on budgetary allocations or any other policy-making, this strategy is costly and time-consuming. In order to assess the economies of those with less resources, as described above, without compromising efficiency, this challenge forced researchers to look for a different methodology.

Over the past few years, the advancement that has been made in deep learning has been enormously used for object detection using satellite imagery for poverty prediction. Deep

learning networks have significantly enhanced object Identifiers making significant advancements in object detection. Artificial intelligence involves machine learning, which essentially entails taking in designs from models or sample data when the machine accesses the information and may benefit from it. A specific kind of machine learning called "deep learning" includes learning in phases. With more commercial companies entering the market and more widespread use of satellite monitoring, improvements. Deep Learning spatial and temporal resolution and poverty prediction bring up new applications, markets, and uses, including the potential to monitor significant sustainability results at scale. Additionally, satellite imagery provides a variety of observations that may be used for socioeconomic development, such as poverty level evaluation, in developing nations without considerable prior data.

Advanced machine learning approaches can carry out complex tasks like object recognition and classification thanks to such extensive and high quality picture data, and deep learning in particular has shown significant promise. While several recent papers have discussed the use of deep learning on satellite imagery for applications in land use cover, urban planning, environmental science, etc., there are still a lot of open questions in the field, particularly when it comes to the use of deep learning for socioeconomic growth.

Therefore, in our paper we will implement deep learning along with socioeconomic studies which involves satellite imagery. In order to determine the degree of poverty using satellite imaging data, we will train Convolutional Neural Network (CNN) models which include (AlexNet, ZFNet, VGG, and ResNet). With the help of this study, it will be possible to create forecasts in unexplored areas, augment the spatial scope of field survey data, and assess socioeconomic status accurately and automatically. Once all the models have been trained, it will then be evaluated using AUROC curve, RMSE score for imagery data, accuracy of the model

and F1 score, precision and confusion matrices. and thereby will be compared based on their performance status.

Literature Survey of Existing Research

In this paper, the author proposed a framework to predict city level poverty. For this we chose an e-commerce dataset and applied machine learning models. Author applied Support Vector Regression (SVR) and Deep Neural Network (DNN) on the datasets and came up with a model. DNN with feature selection performed better than the SVR model in terms of accuracy. This data is not only useful to assess poverty but also to analyze the overall expenditure trend of that city. But this approach has a disadvantage of getting e-commerce transactions or having less number of transactions from some cities. This may affect the predicted results from the actual results (Wijaya et al., 2020).

In this paper, Author considered a survey of households which consists of information related to their financial status like total income, expenditure, child number, no. of T.V, and etc. The datasets contain a total of 47 features. Dataset consists of some impurities which went through the cleansing process. Normalization, Standardization was applied on the dataset inorder to scale-up the values in a range and to make values to center around the mean with unit standard deviation. In the process of this, the dataset was splitted into three subsets that are 90% into the training set, 10% testing set and 10% validation set which is a part of the training set. To handle imbalanced data, techniques called random undersampling, random oversampling, Synthetic Minority Over-Sampling Technique (SMOTE), class weights are applied on the dataset. Author applied 16 machine learning models on this dataset and compared their performance using evaluation metrics. Among all the applied models, Bagged Decision Trees and Light-BGM

performed better than other models with 80% and 81% F1-Score respectively (Alsharkawi et al., 2021).

Food is one of the basic necessities of a person which directly correlates with poverty. In this paper, the author tried to predict the relation between arable land use and poverty. Data was collected from multiple data sources like World Bank, Global Agro-ecological Zones (GAEZ v3.0), Protected Planet. Author explained and collected data of different attributes like Ratio of Actual to Potential Yield (RAPY), Ratio of Actual to Potential Cropland Area (RAPC), Irrigation percentage etc. Using all the collected data related to agriculture and poverty built a machine learning model of Non-parametric CART and random forest. This study reveals the strong relationship between poverty with yielding of crops and land utilization. From the obtained results it can be concluded that the gap in yield relates to poverty. Ratio of Potential Cropland in Protected Areas (RPCPA) is directly proportional to poverty whereas Fertilizer is indirectly proportional to poverty. Apart from this, the built model is able to predict poverty using arable land use with 73.8% accuracy (Tian et al., 2022).

The goal of this study was to use a variety of machine learning models, such as linear regression, decision trees, and random forest gradient boosting models, to forecast poverty in a specific location while accounting for a variety of parameters. With a maximum depth of 7, a maximum number of features of 20, and a maximum number of estimators of 50, the gradient boosting technique predicts poverty with the highest accuracy of 78.5% after grid search and cross validation on each model. Authors also assessed how machine learning models might predict whether or not a person will be poor based on criteria including education, nation, urbanity, age, and technologies. As a result, it will assist the local government and NGOs in creating policies and programs to boost the economy and end poverty (Zixi, 2021).

Strategic policy proposals rely on measuring the decline in poverty and swiftly pinpointing the prevalence of poor areas, especially in those areas where the system is weak.On the contrary side, using data from satellite images may be a more effective technique to assess socioeconomic level. By taking into account 338 villages in China, the authors of this study used machine learning to evaluate poverty at the village level. High quality imaging data has been used to extract significant characteristics. With a 72% accuracy rate, the authors used a random forest model to forecast poverty. The analysis revealed that agricultural output conditions were the least important factor in forecasting a village's level of poverty, with land development and access to utilities and services contributing the most. By employing geospatial data, the author of this study hopes to anticipate poverty at the village level and quickly identify poor places (Hu, 2022).

By using objective functions and stochastic analysis, the author of this study evaluates the accuracy of several econometric models, including OLS and Logit models, along with certain machine learning techniques, random forest and Lasso models, in forecasting poverty at the household level. The models' accuracy is heavily influenced by the poverty rate and the income distribution in a given location. Using AUC and ROC curves, all the models have been evaluated and contrasted. In order to provide more thorough information than the AUROC curve, the author conducts stochastic dominance analysis across the curves. Before selecting the best model for predicting poverty level, all of this was done, and the results showed that the Logit and Lasso models outperformed random forests with an accuracy of 69.43% (Verme 2020).

The author of this study discusses ways to abolish poverty in households. The main obstacle in this study is to identify a way to accurately predict households in poverty. Some ways discussed include asset holdings to predict poverty, however, these indicators are not sufficient

enough and may vary from case to case. Machine learning is one approach to tackle this problem. Data is collected from the Demographic and Health Surveys website (DHS), this website contains survey data for countries and their GPS coordinates. In this study data from Kyrgyzstan is used. Data from Kyrgyzstan is used to predict the accuracy of poverty by applying machine learning models. The first model used is XGBoost, to see how it handles all the variables from the survey data and to test its ability of processing many features. Another model used is the generalized linear model (GLM), which is compared to XGBoost. The results from these models showcased that it is not necessary to have many features to predict poverty, a few important features would also be sufficient for the models to perform well. These results also showcased that XGBoost was more accurate and performed better than the generalized linear model when dealing with priori features (Li et al., 2022).

The author of this study indicates how crucial it is to have reliable data for developing countries, so they get the proper help they need. Lacking reliable data is a major obstacle why most countries have slow economic growth. The author further elaborates that with proper data, countries can receive resources such as food and disaster relief, which would increase sustainable development. Collecting poverty data is labor-intensive, the study recommends using high-resolution satellite imagery data which is becoming more popular and is inexpensive. The author suggests using a transfer learning approach to tackle this obstacle, in which nighttime light data is intensified and the data is richer. This data would be more efficient to fully train a Convolutional Neural Network to predict poverty using nighttime and daytime imagery simultaneously. Lastly the author states that this model would be trained to filter different lands and its physical features and these features from the mode would be more informative to detect poverty on the map rather than using data collected from traditional surveys (Xie et al., 2016).

This study elaborates on how nighttime satellite imagery can help predict which countries are developing. Many countries lack economic development compared to the rest of the world. It also gets too expensive conducting comprehensive surveys for these countries. With satellite data the government can help track these countries and provide them with resources such as food, money and clothes etc.. However, there are some obstacles we face with nighttime imagery data as it contains a lot of noise which makes it difficult to predict. To help combat this problem we would rely on deep learning techniques, the study contains resources where successful deep learning models have been developed such as VGG-Net, Inception-Net, ResNet, and DenseNet to help lower noise and extract more deep features from satellite imagery and use lasso regression to predict poverty. The study further talks about enhancing these models with squeeze and excitation (SE) modules and focal loss, which would tweak the result even better, making the data more accurate. The study concluded that the best result they got was from using the DenseNet model, which was the best performing model out of all the others. The model was also enhanced using the squeeze and excitation module and focal loss which gave even more optimal results (Ni et al., 2020).

In order to improve the accuracy of poverty predictions, this study uses machine learning techniques to investigate the association between nighttime light statistics and poverty. With advancements in machine learning and night light remote sensing, poverty mapping in conjunction with machine learning, night lights, and deprived areas can fill the data gap and serve as a decision-making basis for sustainable development. The goal of the study (Xu et al., 2021) is to predict poverty in the Chinese province of Guizhou using data from nighttime light remote sensing and machine learning techniques. The study's identification accuracy for counties with high levels of poverty was 76.5%.This project intends to predict poverty in long-term time

series county-level areas lacking data by combining DMSP/OLS and NPP/VIIRS night lights data with machine learning and statistical data.

The recent development of machine learning and remote sensing provide a chance to create quicker and less expensive techniques of determining the state of poverty and malnutrition. The study focuses on using publically available, current, and georeferenced data to forecast the prevalence of poverty and malnutrition in the population. Eleven USAID Feed the Future (FTF) priority countries are the subject of the survey. Their overall performance in predicting asset poverty using random forest approaches r2 =.59, which is comparable to previous papers they compared. They also performed feature importance and found that location and remoteness has the highest effect on their model in predicting poverty. The study concluded that the model performance deteriorated when the predictions were made for small sample size, which can be overcome by proper data collection or by using transfer learning techniques (Browne et al., 2021).

In this paper the authors have used a machine learning approach for predicting the use of energy in different parts of Korea. It talks about different poverty indicators available and the pros and cons of each indicator. It discussed objective and subjective indicators. The study focuses on annual data of household income and expenditure surveys done in 2016. Among all the different machine learning models used, the random forest model was chosen as the best model having higher overall performance. Also, the top predictor variables were calculated for the random forest model, the best significant variable was household income. The results of the study are expected to accurately understand the households which are vulnerable to energy poverty and to identify the needs of welfare policies more accurately (Hong & Park, 2021).

In table 2, we describe the approaches methods in the literature surveys we studied and also state the advantages and limitations we found in the studies.

Table 2

Project Deliverables With Descriptions And Dates

Study	**Approached Method**	**Advantages**	**Limitations**
Wijaya et al., (2020)	Linear Regression, SVM, Random forest, Multivariate Linear Regression, Convolutional Neural Network	Used different kinds of datasets with different approaches of models.	Didn't integrated the datasets, analysed them separately, receiving e-commerce purchases or having fewer transactions from certain cities
Alsharkawi et al., (2021)	Applied sixteen different machine learning like Logistic Regression, Ridge Regression, Decision Trees and Light-BGM etc.	Different pre-processing techniques like SMOTE, Sampling, Standardization, one hot encoding	The collected information is expensive.
Tian et al.,	Used Food	Integrated the data	The importance

Study	**Approached Method**	**Advantages**	**Limitations**
(2022)	production dataset and applied Decision trees and random forest	from different sources and derived new parameters like RAPC, RAPY and etc. in the dataset	derived columns are changing with the trend of food production and demand which leads to the effect the relationship with poverty.
Zixi, H. (2021)	linear regression model, decision tree, random forest model, gradient boosting model, and neural network	Defined different evaluation metrics and analyzed to get the most influential features on poverty.	There are not many pre-processing steps mentioned. Their best accuracy obtained is 78.5% with gradient boosting models.
Hu, (2022)	Used satellite images of 337 china villages, classified them using SVM and applied Random forest	Different pre-processing techniques like SMOTE, Sampling, Standardization, one hot encoding	Used R for implementing the methodology, created three categories for poverty.

Study	**Approached Method**	**Advantages**	**Limitations**
Michael Xie et al., (2016)	Applied Fully convolutional Neural Network, VGG	Used night time satellite images with light intensity, Transfer Learning from Image Net.	Night Light image gives only the light intensity which may not be appropriate to estimate the poverty.
Ni et al., (2020).	Applied transfer learning techniques and CNN models	Applied model enhancement using SE module and focal loss	Models used were computationally extensive and requires good resources
Xu et al., (2021)	Predict poverty in the Chinese province of Guizhou	Combined night time images from different satellites	Required statistical data along with the satellite data
Browne et al., (2021)	prevalence of poverty and malnutrition in the population using random forest	Prediction score was good and focused on large number of countries	Requires more data and transfer learning to perform better on small scale
Hong & Park, (2021)	machine learning approach for	Good feature selection and	Used only statistical data and no deep

Study	**Approached Method**	**Advantages**	**Limitations**
	predicting the use of energy in different parts of Korea	importance explanation	learning techniques

*Note.*Summary for literature surveys.

Data And Project Management Plan

Data Management Plan

Our project will consist of text data and image data, thus a solid data management strategy is essential. The plan for managing data will include data collection, management, storage, and, finally, utilization mechanisms. Collecting the data, which would include text data from our surveys and satellite data, would be the first stage. The daytime photos used in the real-time image data come from Google's Map Static API and can be delivered in any size up to 640 by 640 pixels. To help us estimate poverty at night, the Earth Observation Group has also provided us with nighttime imaging data.

It would be challenging to handle and retain the data for our project on the local system because it is gathered from many sources, such as satellite picture, survey, and GPS data. There may be major problems with storing data on a local system. Loss of data, losing your laptop, and exposure to outsiders are a few of these problems. To tackle this problem we will leverage Amazon Web Services. AWS is a popular cloud platform used by many businesses for their applications and infrastructure. The first step is to create an S3 bucket, which serves as a cloud storage object for our data lake

Amazon S3 provides a variety of tools to effectively organize and manage data according to the specific needs of our project. We need a method to monitor expenses since our project involves several data sources. We will use AWS Trusted Advisor to control expenses, a program that supports users in adopting advised procedures to help keep S3 expenses low. AWS will keep our data safe since it enforces basic security measures and encryption to prevent unauthorized access to our S3 bucket. This is why our team chose AWS S3 to manage our data; storage is also not an issue either, since a single bucket can contain up to 5 terabytes. Next step would be to

make the S3 bucket public, meaning any service can access this data in AWS. For services to talk to each other, we need to create an IAM role, which stands for IAM identity. This will allow services to communicate with each other.

As we finish configuring the AWS pipeline for our project data, we will then upload all the different types of data from various sources into the S3 bucket. AWS credentials would be distributed to team members so that everyone could quickly access the data for data management. To manage this data better, we would use another service by AWS called Glue. Glue contains automated python scripts which removes duplicates and missing values in the preprocessing of our data.The data is then stored back in the S3 bucket and is ready to be used. To visualize this data, we would store it in AWS Redshift, which is a data warehouse. From there we can connect it to tableau using Redshift's API and tableau will be populated with data from Redshift.

Amazon SageMaker is an environment to perform machine learning in the cloud. Developers can use this to train their models and deploy them in the cloud. Now that we have preprocessed our data within Glue, we can employ it to develop our models. Initially, we need to create a notebook instance inside AWS and specify the instance size based on the dataset. If the dataset is large, a large instance with increased compute power is required. However, since our data falls within the medium range, we will select a medium sized instance to keep costs low. Once the instance is configured, we can launch the Jupyter Notebook. From there we simply create a notebook and begin working on the model. AWS has its own Jupyter Notebook embedded in the cloud, as mentioned before. Since we have made the S3 bucket public we need to specify the path to the S3 bucket enabling the notebook to access the dataset.

Project Development Methodology

The Cross-Industry Standard Process for Data Mining Methodology (CRISP-DM) is being used in our project for project development. All of the crucial project development milestones are covered by the six primary phases of this technique. CRISP-DM is well known for creating long-term project development strategies with good flexibility. Any project involving data science can use this template.

Business Understanding

The first phase of CRISP-DM, which involves understanding the project in the terms of the business world. In the course of doing that, we come to realize how crucial poverty is to a nation's ability to prosper economically. Government agencies will develop strategies and plan on how to reallocate resources to different areas based on poverty level. Assessing poverty through traditional surveys is expensive, time-consuming, and requires a lot of human resources. Machine learning can be used as an alternative to evaluate poverty, offering a more efficient approach. This stage greatly aids in our ability to comprehend the project's history.

Data Understanding

The first is survey information used to calculate the poverty index, while the second is location-specific satellite imagery. Images data is a little more difficult to work with than survey data, which is straightforward tabular data of text and numbers. From the Demographic and Health Surveys (DHS) website, we gathered survey information. There are dozens of unnecessary columns in this survey. Only the relevant columns from the economic sector will be filtered. Both daytime and nighttime satellite photos of a location are required for the deep learning model. We intend to get daytime photos via the Google Static Map API, and nighttime images from the same DHS data source. We must make a valid request, which must include our

abstract, in order to acquire DHS data. They will grant access to datasets after assessing the request.

A single nighttime image takes up between 400 and 500 MB, which is a significant amount of space on a local machine. For the purpose of handling and storing the data, we are using Amazon AWS. We will segment the daytime and nighttime photographs after gathering them using chosen coordinates that correspond to certain locations and make model training simple. To better comprehend the data and to help uncover any blind errors or flaws in datasets, we will create an initial insight report after gathering all the data.

Data Preparation

One of the critical steps in getting a dataset suitable for model training is this phase. Any significant faults or difficulties with the dataset will be found and addressed in that report. In order to do that, we will follow a plan for preparing the data, which includes handling missing values, outliers, different data formats, etc. We shall employ appropriate approaches for each problem to address these anomalies. Dealing with the image dataset presents the main problem for our study. Daytime RGB images and nighttime RGB images are the two different types of images we have. Although images have a high resolution, we will resize them to 640 by 640 pixels. While daytime photographs require a method to remove clouds, nighttime images downloaded from the DHS website are free of clouds.

We will utilize Python libraries like earthy, numpy, array, and matplotlib to remove clouds from satellite photos. We may eliminate any clouds or shadows from the image using these tools. Following these preprocessing steps, we will pair up the daylight and nighttime photos. In other words, the photos will be divided up and given a location name. After that, using clustering techniques like K-means, FCM-fuzzy C-mean algorithm, and others, we will create

clusters in the daylight photos to identify various characteristics of the image, such as highways, river bodies, buildings, and so on. The photos are now prepared for model training after being clustered.

Modeling

After preprocessing, the data is free of all kinds of anomalies and acceptable for model training. We will choose a model that works with our problem set during this phase. Regression models are required for survey data, and deep learning models are required for picture data. The entire dataset will first be divided into three smaller datasets for training, testing, and validation. Training dataset makes up 70% of the total, while testing and validation datasets each make up 15%. We will now choose the best algorithms for our dataset. In this project, we'll use deep learning Convolutional Neural Network (CNN) models like AlexNet, ZFNet, VGG, and ResNet to calculate the poverty index. Python packages are utilized to implement these methods. These algorithms are all pre-structured and have a lot of parameters. To improve performance, factors like filter size, the quantity of hidden layers, and the thickness of those layers (measured in neurons) are crucial. Some parameters can be changed, while others are fixed.

Evaluation

The purpose of this step is to assess the models that were created in the previous phase. Two models are involved in our project: one is for survey data that needs to be used to calculate the poverty index, and the other is for deep learning models. For the purpose of performance cross-checking, both models must be assessed. Various evaluation techniques, including Mean Error (ME), Mean Absolute Error (MAE), Mean Percentage Error (MPE), and Root Mean Square Error, will be used (RMSE). These metrics are mathematical formulas that, in various methods, compute the discrepancy between the actual expected value and the value received

from the model. These measurements will be used to help us choose the most effective models for our issue.

Deployment

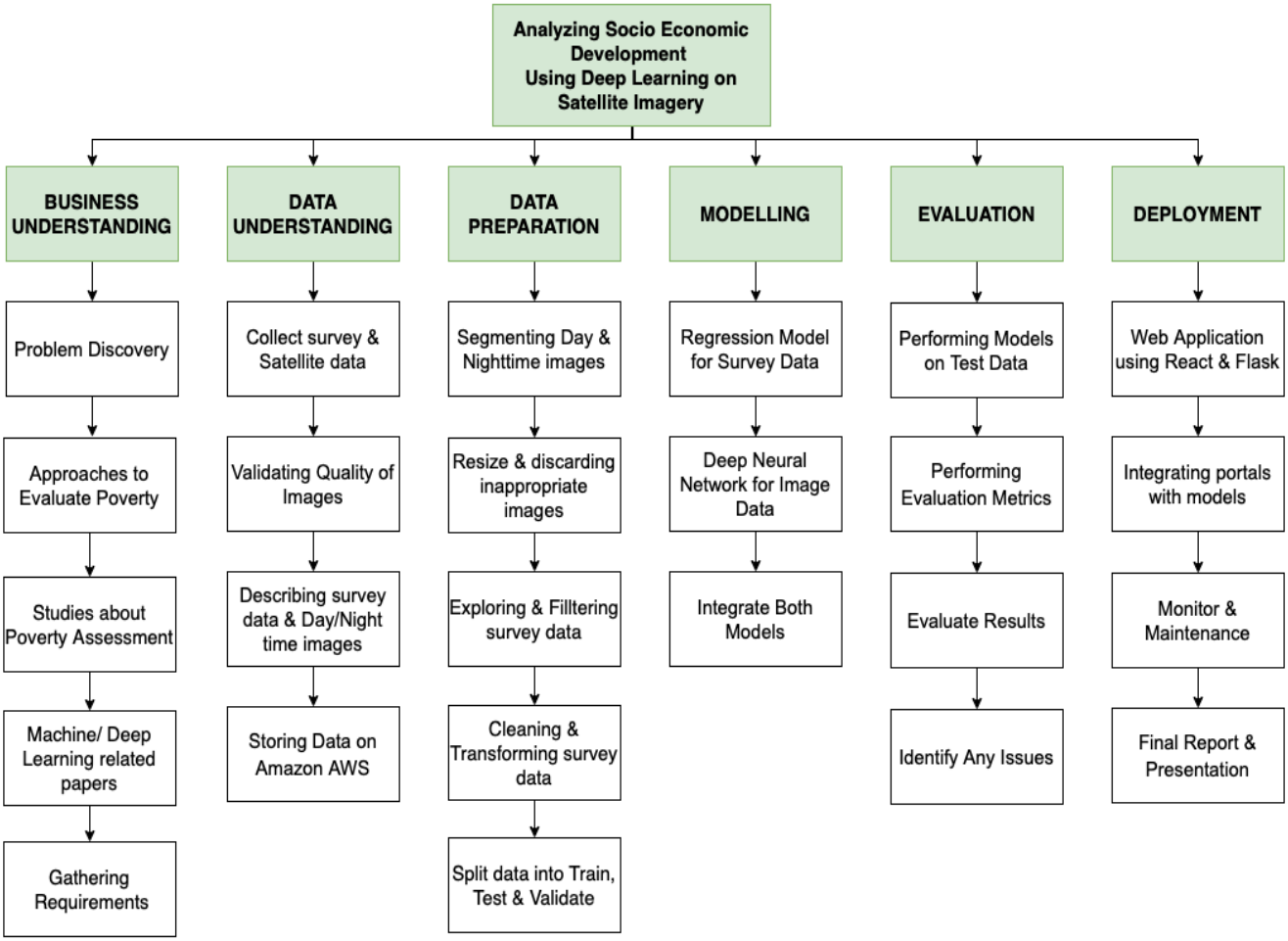
Utilizing Amazon AWS, a web application will be created and hosted. The application will be created to satisfy all of the website's requirements, including registration, logins, email, and others. On that website, you can find all the insights from the dataset as well as information about the project, including the models that were used, the datasets' sources, their performance, and more. The website will implement the developed deep learning model. A website provides details on the poverty index as it is shown on a map. In addition, a unique function will be offered, giving users the chance to submit any satellite image of the location. After uploading, a satellite image model will evaluate the image and forecast the area's poverty index.

Project Organization Plan

To develop the Work Breakdown Structure for this project, we are using the CRISP-DM methodology (WBS). The approach, as seen in the figure 1, involves six essential steps that collectively characterize the project's entire workflow. The first and most crucial stage is Business Understanding, which focuses on describing the issue that this study is seeking to address by anticipating poverty using deep learning algorithms and satellite imagery data. A key component of economic development is researching numerous models to determine how to measure poverty in the modern period. The primary goal is to create a model that uses deep learning and data from satellite imaging to predict the socioeconomic standing of a particular region as this model will help the government reassess its policies and laws in order to improve the economy and end poverty.

Figure 1

Work Break-Down Structure



The following step in this methodology is Data Understanding, which focuses on the conclusions we can draw from the raw data and what we may anticipate will help us accomplish our goal. We require high resolution satellite images of the area and economic survey data from the LSMS and DHS websites in order to determine local poverty. The following process, known as Data Preparation, involves actions to clean, resize, format, visualize, and do feature engineering on the final data. The raw data will be put into Amazon SageMaker, where all of these stages will be carried out. Our image size is 400 × 400 pixels, thus the first step in this phase is to segment the day and night time images. We'll also try downsampling to find the

optimal size image to train our model because the image size is too large. In order to improve the model's performance, normalization will also be carried out.

Both the survey data from LSMS and DMS as well as the imaging data from NOAA will be available once the final dataset is complete. On the basis of satellite night light data, deep learning models like CNN and ResNet will be used to anticipate poverty as shown in the DHS survey data, on which machine learning models have previously been applied. evaluating the ability to extract features from daylight photographs using deep learning frameworks in order to forecast wealth. A heatmap that displays the distribution of the expected poverty level for a certain place will be created when all the models have been run on the satellite and survey data.

The evaluation stage, the fifth phase, involves testing the effectiveness of machine learning models applied to survey data using the AUROC curve, RMSE score for satellite images, accuracy, f1 score, precision, and confusion matrices. These results will be used to evaluate how well a model predicts the degree of poverty. Deployment is the sixth and final stage, where all the models and data are combined on a web application using React and Flask and a heatmap showing the degree of poverty in a specific location. However, monitoring and upkeep will also be used to track the efficiency of the algorithms for predicting poverty. Creating a project report and presentation detailing every step of the process as well as the overall project plan.

Project Resource Requirements & Plan

The entirety of your project will be conducted on a local machine, so it's crucial to have a system that can maintain the project. It’s recommended that the system has a 64 bit architecture to provide high compute power to run heavy applications, and the RAM should also be 8 GB or higher to support many applications running at the same time. Since the majority of our

teammates own the most recent model of MacBook Pro with the M1 processor, which uses 128 bit, we have more than enough processing capacity to perform the large datasets required for our project.

We will initially keep our data on the local system and test the code's functionality using Annoconado's Jupyter notebook. Our plan is that the data would be kept on Amazon's cloud infrastructure in places like the S3 data lake. Data storage on the local system is problematic for our project, as we specified in our data management plan. With its high latency and availability throughout its regions, S3 will guarantee that our data is safe. Other AWS services can connect and access the data in the secured S3 bucket by using IAM roles. S3 storage costs $0.023 per GB per month. Although this seems inexpensive, the price does stack up over time. Each region has different price tiers, our team will use the Northern California region. Project data will remain in the S3 till the end of our project in May 2023.

The majority of our datasets are made up of image data, whereas our survey data is made up of text data with a large number of missing values. To ensure our project is successful we need the correct resource requirements. Python is our first requirement and will be used throughout the project as our primary programming language to handle missing data and outliers. Python is a user-friendly, open-source programming language. Throughout the world, data scientists work with data using this language to assist them solve complex problems. The most recent version of Python 3.10, will be utilized for our project, along with all of its libraries and frameworks. Python also has a number of libraries, which is a benefit. Our team will mostly use Pandas, Matplotlib, and Scikit-learn libraries. Pandas will assist with data manipulation, including importing, analyzing, and altering our data. We may learn from our data by displaying it in different graphs and scatter plots using Matplotlib. Finally, scikit-learn is used to execute

machine learning, which includes clustering, classification, regression, and other algorithms to work on various data-related problems. Using the pip install commands, these libraries can be downloaded for no cost.

These libraries will be in a Jupyter notebook and will also demand considerable computing power for our project. Jupyter notebook has an easy to use user interface and is highly compatible with Python. Jupyter notebook can be used both locally and in the cloud. Anaconda is a data science tool that runs locally and includes Jupyter, and AWS includes Jupyter as part of SageMaker. No matter where jupyter is used it will use the system ram to run our complex models. Jupyter notebook is open source and free to use, however running it on a SageMaker instance does cost a small amount depending on which size of instance is used.

Our project's documentation, including reports and chapters, will be done through Google Drive. Resources such as Google Slides will also be used for presentations. We utilize Google Drive services as they allow us to collaborate and since everything is stored in the cloud, where it is safe. To ensure the effectiveness of any project, it is crucial to have a solid project management plan. We will manage our project, assign tasks, and monitor project status using Jira, a well-known project management platform. Work breakdown structures (WBS), gantt charts, and other planning tools are only a few of the numerous features that Jira offers. Jira is free software that works well for our project.

Table 3

Resources With Cost Estimation

Resource Function	**Resource Type**	**Resource**	**Duration**	**Cost**
Local System	Hardware	128-bit Version machine or later, 8 GB RAM, 256 GB SSD	09/09/2022 - 05/10/2023	Free
Machine Learning Platform	Software	Scikit - learn within SageMaker	10/03/2022 - 05/10/2023	Free
Amazon Web Services	Platform	AWS Services	11/14/2022 -05/10/2023	$ 29.00 per month
IDE	Environment	Jupyter Notebook	09/15/2022 - 05/10/2023	Free
Python Libraries	Software	Pandas, Matplotlib, Scikit-learn	10/03/2022 - 05/10/2023	Free
Programming Language	Software	Python (3.10)	10/03/2022 - 05/10/2023	Free
Documentation	Software	Google Docs	08/29/2022 - 05/10/2023	Free

Resource Function	Resource Type	Resource	Duration	Cost
Presentation	Software	Google Slides	08/29/2022 - 05/10/2023	Free
Project Management Tool	Software	Jira	09/15/2022 - 05/10/2023	Free
Total			244 days	$ 116-150

Note. Table showing the prices of required resources for the project

Project Schedule

Gantt Chart

The project development process includes the phase of project management. It serves as the project plan's overview and aids in breaking the project up into manageable pieces to be finished within a set amount of time. We can plan projects and monitor their progress with the use of Gantt charts. With the start and end dates of each smaller segment of the project and the specific person allocated to the work, it provides us with a visual representation of the project plan. Additionally, it shows us how the entire project will proceed. It also demonstrates the course of the entire project. Figure 2 shows the gantt chart task and people responsible for the tasks and timelines for each task.

Figure 2

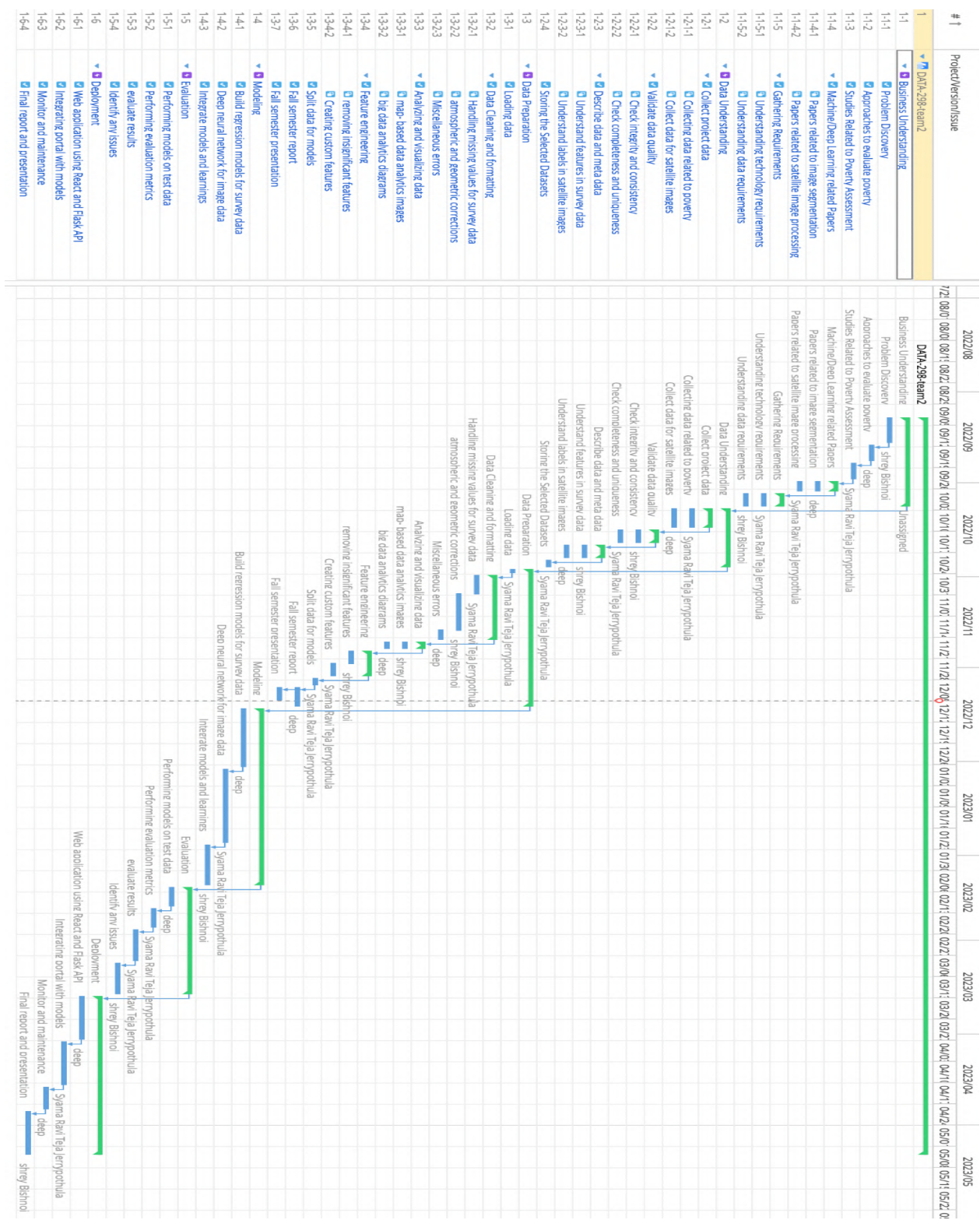
Gantt Chart (Part-a)

#	Project/Version/Issue	Responsible	Responsible	Start ↑	Finish	Duration
1	DATA-298-team2			2022/09/09	2023/05/10	244 days
1-1	Business Understanding			2022/09/09	2022/10/08	30 days
1-1-1	Problem Discovery	ahaz.bhatti, deep, shrey Bishnoi, Syama Ravi ...		2022/09/09	2022/09/17	9 days
1-1-2	Approaches to evaluate poverty	ahaz.bhatti, deep		2022/09/18	2022/09/23	6 days
1-1-3	Studies Related to Poverty Assessment	shrey Bishnoi, Syama Ravi Teja Jerrypothula		2022/09/24	2022/09/29	6 days
1-1-4	Machine/Deep Learning related Papers			2022/09/30	2022/10/03	4 days
1-1-4-1	Papers related to image segmentation		deep, shrey Bishnoi	2022/09/30	2022/10/03	4 days
1-1-4-2	Papers related to satellite image processing		ahaz.bhatti, Syama Ravi Teja Jerrypothula	2022/09/30	2022/10/03	4 days
1-1-5	Gathering Requirements			2022/10/04	2022/10/08	5 days
1-1-5-1	Understanding technology requirements		ahaz.bhatti, Syama Ravi Teja Jerrypothula	2022/10/04	2022/10/08	5 days
1-1-5-2	Understanding data requirements		deep, shrey Bishnoi	2022/10/04	2022/10/08	5 days
1-2	Data Understanding			2022/10/09	2022/10/28	20 days
1-2-1	Collect project data			2022/10/09	2022/10/15	7 days
1-2-1-1	Collecting data related to poverty		shrey Bishnoi, Syama Ravi Teja Jerrypothula	2022/10/09	2022/10/15	7 days
1-2-1-2	Collect data for satellite images		ahaz.bhatti, deep	2022/10/09	2022/10/15	7 days
1-2-2	Validate data quality			2022/10/16	2022/10/20	5 days
1-2-2-1	Check integrity and consistency		deep, shrey Bishnoi	2022/10/16	2022/10/20	5 days
1-2-2-2	Check completeness and uniqueness		ahaz.bhatti, Syama Ravi Teja Jerrypothula	2022/10/16	2022/10/20	5 days
1-2-3	Describe data and meta data			2022/10/21	2022/10/25	5 days
1-2-3-1	Understand features in survey data		shrey Bishnoi, Syama Ravi Teja Jerrypothula	2022/10/21	2022/10/25	5 days
1-2-3-2	Understand labels in satellite images		ahaz.bhatti, deep	2022/10/21	2022/10/25	5 days
1-2-4	Storing the Selected Datasets	ahaz.bhatti, Syama Ravi Teja Jerrypothula		2022/10/26	2022/10/28	3 days
1-3	Data Preparation			2022/10/29	2022/12/13	46 days
1-3-1	Loading data	ahaz.bhatti, Syama Ravi Teja Jerrypothula		2022/10/29	2022/10/30	2 days
1-3-2	Data Cleaning and formatting			2022/10/31	2022/11/21	22 days
1-3-2-1	Handling missing values for survey data		ahaz.bhatti, Syama Ravi Teja Jerrypothula	2022/10/31	2022/11/06	7 days
1-3-2-2	atmospheric and geometric corrections		deep, shrey Bishnoi	2022/11/06	2022/11/18	13 days
1-3-2-3	Miscellaneous errors		deep, Syama Ravi Teja Jerrypothula	2022/11/18	2022/11/21	4 days
1-3-3	Analyzing and visualizing data			2022/11/22	2022/11/24	3 days
1-3-3-1	map- based data analytics images		shrey Bishnoi, Syama Ravi Teja Jerrypothula	2022/11/22	2022/11/24	3 days
1-3-3-2	big data analytics diagrams		ahaz.bhatti, deep	2022/11/22	2022/11/24	3 days
1-3-4	Feature engineering			2022/11/25	2022/12/03	9 days
1-3-4-1	removing insignificant features		deep, shrey Bishnoi	2022/11/25	2022/11/29	5 days
1-3-4-2	Creating custom features		ahaz.bhatti, Syama Ravi Teja Jerrypothula	2022/11/29	2022/12/03	5 days
1-3-5	Split data for models	deep, Syama Ravi Teja Jerrypothula		2022/12/04	2022/12/06	3 days
1-3-6	Fall semester report	ahaz.bhatti, deep, shrey Bishnoi, Syama Ravi ...		2022/12/07	2022/12/13	7 days
1-3-7	Fall semester presentation	ahaz.bhatti, deep, shrey Bishnoi, Syama Ravi ...		2022/12/07	2022/12/11	5 days
1-4	Modeling			2022/12/14	2023/02/10	59 days
1-4-1	Build regression models for survey data	ahaz.bhatti, deep, shrey Bishnoi, Syama Ravi ...		2022/12/14	2023/01/02	20 days
1-4-2	Deep neural network for image data	ahaz.bhatti, deep, shrey Bishnoi, Syama Ravi ...		2023/01/03	2023/01/27	25 days
1-4-3	Integrate models and learnings	deep, Syama Ravi Teja Jerrypothula		2023/01/28	2023/02/10	14 days
1-5	Evaluation			2023/02/11	2023/03/18	36 days
1-5-1	Performing models on test data	deep, shrey Bishnoi		2023/02/11	2023/02/17	7 days
1-5-2	Performing evaluation metrics	ahaz.bhatti, Syama Ravi Teja Jerrypothula		2023/02/18	2023/02/24	7 days
1-5-3	evaluate results	ahaz.bhatti, Syama Ravi Teja Jerrypothula		2023/02/25	2023/03/07	11 days
1-5-4	Identify any issues	deep, shrey Bishnoi		2023/03/08	2023/03/18	11 days
1-6	Deployment			2023/03/19	2023/05/10	53 days
1-6-1	Web application using React and Flask API	deep, Syama Ravi Teja Jerrypothula		2023/03/19	2023/04/02	15 days
1-6-2	Integrating portal with models	ahaz.bhatti, shrey Bishnoi		2023/04/03	2023/04/17	15 days
1-6-3	Monitor and maintenance	ahaz.bhatti, deep		2023/04/18	2023/04/25	8 days
1-6-4	Final report and presentation	ahaz.bhatti, deep, shrey Bishnoi, Syama Ravi ...		2023/04/26	2023/05/10	15 days

This project will last for eight months, from September 9, 2022, to May 10, 2023. For each task there are at least two people assigned and for some of the complex tasks three people are assigned. Figure 3 shows the full gantt chart with the project divided into multiple tasks which has many smaller subtasks.

Figure 3

Gantt Chart(Part-b)



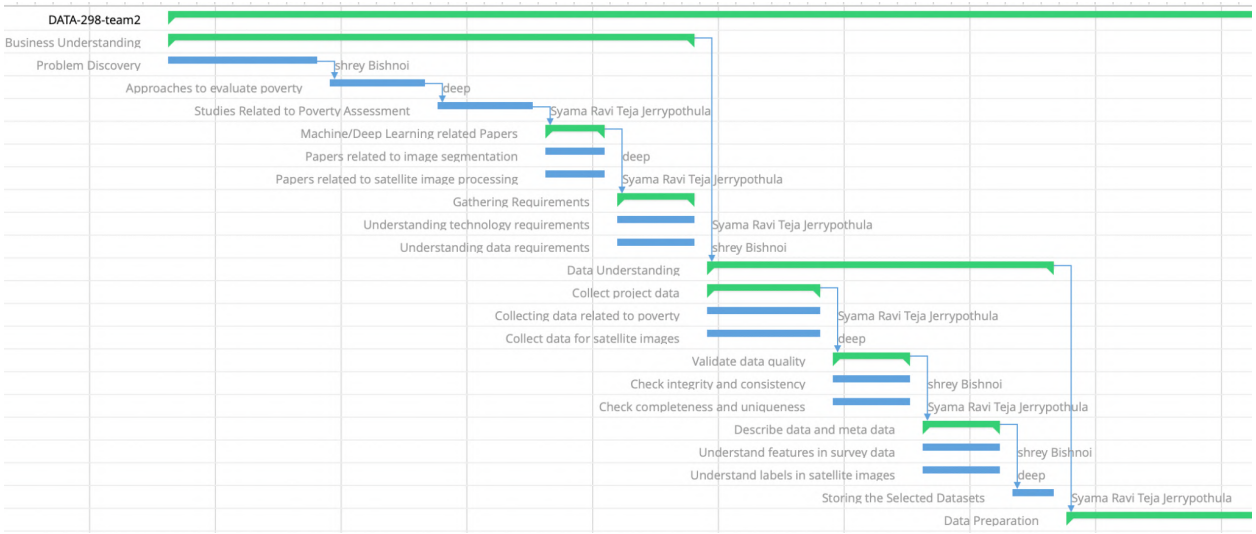
The project understanding phase as shown in the figure 4, is divided into multiple subtasks, where problem discovery is the first task in this project where all the team members are assigned. The team will be able to gain a sense of direction and a plan of action by studying various ways that are relevant to this project. As seen in the image, two team members are tasked with researching studies that evaluate poverty, while the other two are given the task of studying the existing works linked to poverty evaluations. The following goal in this phase is to learn more about the deep learning techniques that can be employed for this project, with subtasks that include reading up on publications on picture segmentation and satellite image processing.

Upon the completion of studying research papers and related works. We begin by gathering the project's needs, which is a crucial step in any project. We think of this stage as a two-part procedure. The first step is the gathering of the necessary technology, for which two people are allocated. The second step is the gathering of the necessary data, for which two individuals were also assigned. We must comprehend both the need for satellite data and the need for survey data in order to meet the data requirements.

The next stage is data understanding, which comes after learning about the project needs. We must gather survey data that includes information gathered by authorities from various nations, as well as satellite data with daytime and nighttime photographs for various places. After collecting the data, verify its accuracy. Each person has been assigned a specific task to perform, which includes examining the data's consistency, accuracy, completeness, and uniqueness. Describing the data that will provide labels for the satellite images and features in the survey data.

Figure 4

Gantt Chart (Part-c)

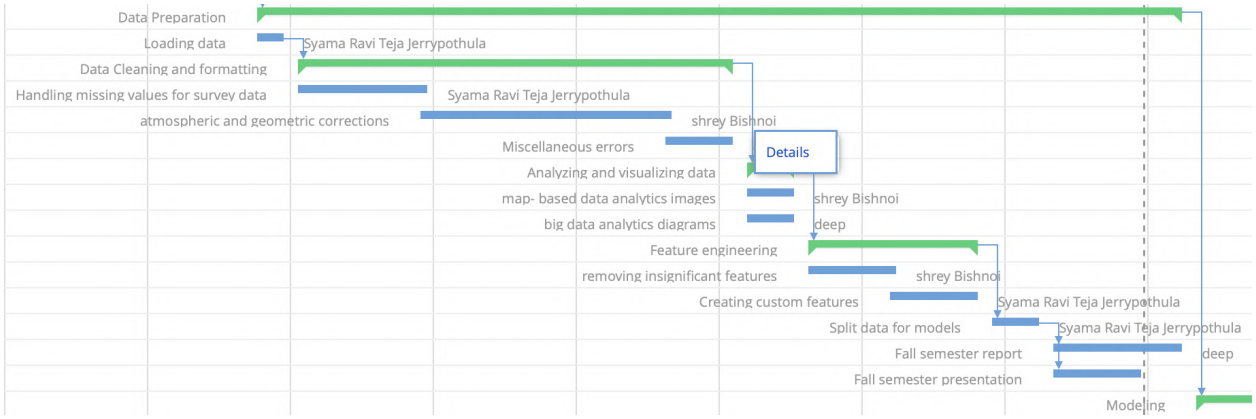


The following data preparation period will run from October 29, 2022, through December 12, 2022. The data preparation process involves numerous processes. The procedures involve importing the data, cleaning the dataset to remove missing numbers, abnormalities, and other types of mistakes, and addressing them. We will apply atmospheric, geometric, and other associated modifications to the satellite photos.

Once the data has been cleaned and processed, we will further analyze it and produce visuals using big data analytics diagrams and map-based data analytics images to help further explain the data. The following phase is feature engineering, which is crucial for any machine learning or deep learning project. In this step, we'll find and eliminate the elements that aren't important and add certain features that are necessary for the modeling stage. After finishing the feature engineering task, the data will be given on to be split for the models. There are 3 categories: training, testing, and validation for the data. After this task is finished, the data preparation phase will come to a conclusion. Figure 5 shows the data preparation phase for the project.

Figure 5

Gantt Chart (Part-d)



Additionally, at the conclusion of the fall semester, we will provide a report and presentation outlining the project's progress. The report will cover all of the work completed up to the point of data preparation.

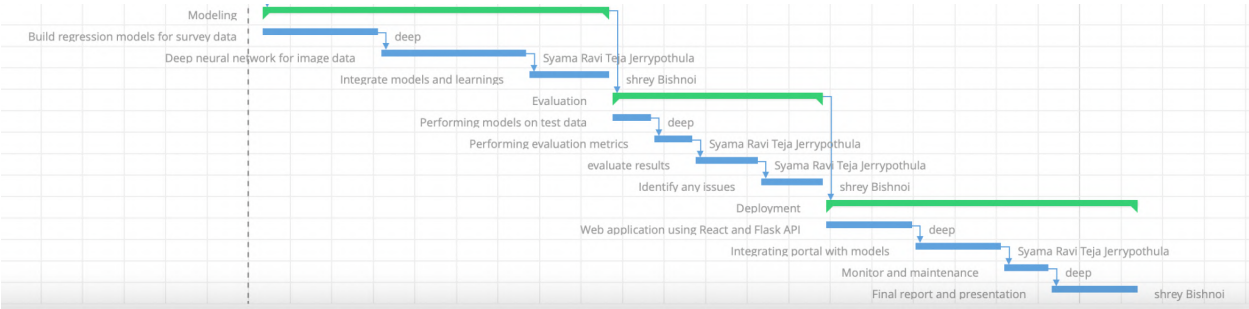
Modeling comes next, and it has a number of related activities. This phase of the project will involve creating regression models for the survey data and deep learning models for the satellite data. Regression and deep learning results will be combined into a single entity once the models have been developed.

The following step, which will take place 36 days after the modeling phase, will involve evaluating the models. The evaluation of the models is a crucial stage because it reveals how well the models function and enables us to choose the model that performs the best. The models will be used with test data, and evaluation metrics displaying the performances will be produced. Models will be assessed using the metrics, and any additional problems will be found before any changes to the models are made.

The next phase as shown in the figure 6, is deployment, where the models will be deployed on the web portal. Utilizing React and the Flask API, we will build a web application for deployment, and the following subtask is to integrate the models with the finished portal.

Figure 6

Gantt Chart (Part-e)

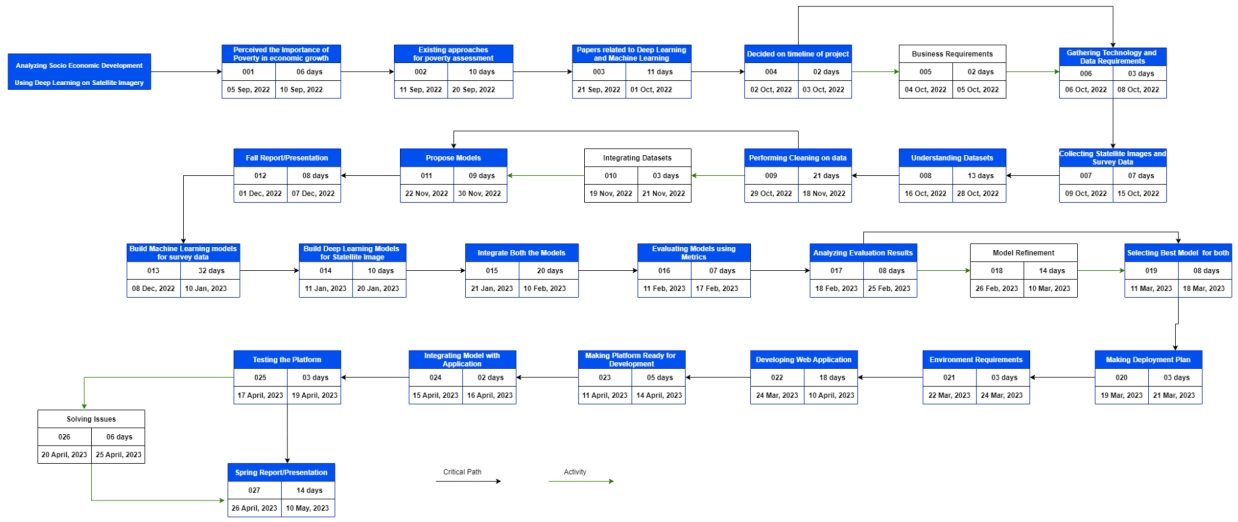


Pert Chart

A pert chart is a project management tool that highlights the crucial phases of a project. The milestone-based and task-based pert charts are the two different forms of pert charts. The longest path, which is also the critical path, lists the steps that are absolutely necessary to complete the project, along with how they relate to the other activities. So, the critical path will undoubtedly be implemented. The non-critical path includes the steps on which a project's completion is not reliant; that is, the project can still be completed without completing the non-critical activities.

Figure 7

Pert Chart



The project's initial step, Analyzing Socio Economic Development Using Deep Learning on Satellite Imagery, is seen in figure 7. Understanding the impact of poverty on economic development is necessary. Researching current strategies will be the first step in the following phase, which will aid in understanding a variety of pertinent strategies to provide the project with a framework and overview.

Planning the project is important since without it, we can't move forward to the end of the project, including choosing a timeframe and the stages to take. The following phase is business knowledge, which aids in defining the project's business utilization. This step is optional and not necessary, thus it can be avoided. The following stage is to compile the data and technical requirements that the project will require. Data will be gathered when the requirements have been gathered, and it will be examined to better understand and clean it. Moving immediately to the following stage of selecting the models can be done without skipping the subsequent step of integrating the data, which is not necessary. Once the models have been developed and reviewed using evaluation criteria, we can modify the models or move straight to choosing the top models

because neither will delay project completion. The project deployment's future phases, final report, and project presentations are all outlined. Useful charts include the pert chart, Gantt chart, and WBS. For the same project, each chart presents information in a different way. After carefully planning everything, proceed by efficiently executing the duties.

Data Engineering

Data Process

This project achieves its goal by utilizing a range of datasets and types. According to the project criteria, we require data from satellite images, as well as survey data from homes to assess their wealth. We require both night and day images from satellites. Data requirements are stated in table 4, along with the purpose of these datasets in relation to the project. For this project we chose a country called Rwanda, which has a land size of 26,338 square kilometers and is organized into 30 administrative districts (Wikipedia contributors, 2022). To acquire raw survey data for Rwanda, our team made certain that we diligently searched for a reputable website in order to obtain the most reliable findings. Since demographic data is confidential we must locate a credible website. We identified several online sources such as The Demographic and Health Surveys (DHS) website (The DHS Program - Available Datasets, n.d.) and Datarade website (Cloudflare, n.d.). These websites have collected, analyzed, and shared credible demographic data. Another website we discovered was the United States Census Bureau; however, this website only offered demographic data for the United States.

The next step was to locate a reliable source for nighttime image data that would display night light intensity across the world. Many websites included nighttime images, such as Earthdata from Nasa, which included cloudy nighttime images (Hall, 2022). Following that came the Citiesatnight website, which featured individual nighttime images of countries rather than the entire world (Gallery, 2022). Another resource we looked at was the National Oceanic and Atmospheric Administration (NOAA), a government website headquartered in the United States that includes high quality night light intensity images of the world (n.d). Now that we have several data sources to assist us forecast wealth using nighttime images, we want to investigate if

daylight images deliver a better result. Just like nighttime images, there were many sources for daytime images. The first source we identified was Earthexplorer, which featured daylight images affected by clouds (Survey, n.d.). Another source we discovered was Google Static Maps, which contained the most accurate and up to date daytime images with minimal clouds (Maps Static API, n.d.).

Once we have finalized the data sources for all types of data, we will collect the data and do preliminary data exploration on the datasets such as metadata about the dataset, distribution plots, box plots, basic statistical metrics such as mean, mode, median, missing data percent, and so on. This stage will help in the identification of flaws in the dataset. After performing the necessary actions on the dataset, we will do some statistical comparison analysis to get key takeaways that will aid in the selection of models to apply to datasets. After preprocessing the data, we want to divide it into 70-10-20 chunks, with 70% training data, 10% validation data, and 20% testing data, and then apply machine learning models to them.

Table 4

Required Data With its Purpose

S. No	**Data Requirement**	**Purpose**
1.	Satellite night lights	Satellite image image file giving nightlights intensity around the world.
2.	Survey data	Household survey data to provide indicators in the areas of population, health, and nutrition.
3.	Daytime satellite imagery	Predict wealth using daytime imagery.

Data Collection

We started exploring the possible sources where we could get the data related to our project. We also assess the quality of the data as part of this process. Our team also tested all the dataset sources with our code and validated the ones that we had most success with. The details of the datasets we picked are described further below.

Household Survey Data

We chose DHS for survey data over USAID since it lacked quality and quantity while analyzing. To gain access to the data on the DHS website, we submitted an abstract using an open forum provided on the website, following approval, we were granted access to the data. The DHS website includes a broad variety of data to deal with. After understanding the acronyms of the data, we selected survey data listed under the section household recode, with the filename RWHR70FL in zip format for the year 2014-2015.

This zip file contains nine files in various formats with various information. However, we simply require one file in data file format. This file contains 12,699 rows of data from household surveys and hundreds of columns. We simply need a few columns from the file that are related to cluster and wealth information. We referred to the DHS-VII Recode Manual document to find the correct columns related to them (DHS Recode Manual (English), n.d.). Page 15 of the document contains information regarding the columns. A sample of the data was shown in figure 8 using Python.

Figure 8

Data File Containing Wealth Index

```
1 1RW6      1   1  1   1 10039641120141379 6 1 0 6 6 11151 41 10   1  1414 22 0        0 41 42 1 11776 11 13    5310000000112134 44 11530  810100
1 2RW6      1   2  1   1 10039641120141379 3 1 1 3 3 01171 47 10   1  1414 22 1 1003668 41 42 1 01776 20113   10211110000343531 22 112800 71 1110   0   13 1100000     00
1 3RW6      1   3  1   1 10039641120141379 1 0 0 1 1 01181 46 10   1  1414 22 0        0 41 42 1 01776 01 13    3231000100343531 11 112701 71 1110   0   13 0000000     00
1 4RW6      1   4  1   1 10039641120141379 5 4 0 5 5 01171 45 10   1  1414 22 1 1003668 41 42 1 01776 40113    5210000000112134 25 124400 81 2          13 0000000     00
1 5RW6      1   5  1   1 10039641120141379 4 1 0 4 4 21151 45 10   1  1414 22 0        0 41 42 1 21776 11 13   10210000000112134 12 113300 8111110   0   13 0000000     00
1 6RW6      1   6  3   1 10039641120141379 8 3 0 8 8 01171 44 10   1  1414 22 0        0 41 42 1 01776 31 13   15221110110343134 34 115000 81 2          13 1100000     00
1 7RW6      1   7  1   1 10039641120141379 2 0 0 2 2 01151 42 10   1  1414 22 1 1003668 41 42 1 01776 00141   60220100000112134 41 128000 81 2          13 1100000     00
1 8RW6      1   8  2   1 10039641120141379 6 1 1 6 6 01171 47 10   1  1414 22 1 1003668 41 42 1 01776 20113   30220000100113534 22 113700 81 1110   0   13 1100000     00
1 9RW6      1   9  2   1 10039641120141379 6 1 3 4 6 01171 45 10   1  1414 22 1 1003668 41 42 1 01776 40141   10210000000112134 34 125400 8101101   0   33 1100000     00
110RW6      1  10  2   1 10039641120141379 4 2 0 4 4 01161 46 10   1  1414 22 0        0 41 42 1 01776 21 13    8211100100343534 14 114401 80 1010   0   33 1010000     00
111RW6      1  11  1   1 10039641120141379 4 1 0 4 3 01171 44 10   1  1414 22 0        0 41 42 1 01776 11 41   30230000000112131 21 124600 81 2          13 1100000     00
112RW6      1  12  2   1 10039641120141379 6 2 0 6 5 11151 44 10   1  1414 22 0        0 41 42 1 11776 21 41   30220100100343534 15 113700 8112          13 1110000     00
113RW6      1  13  2   1 10039641120141379 4 2 0 3 4 01171 45 10   1  1414 22 0        0 41 42 1 11776 21 13    5210100000343534 24 125700 8111101   0   13 1110000     00
114RW6      1  14  1   1 10039641120141379 1 0 0 1 1 01171 45 10   1  1414 22 1 1003668 41 42 1 01776 00113    5220100000113534 11 126100 81 2          13 1100000     00
115RW6      1  15  2   1 10039641120141379 5 1 0 5 5 11151 41 10   1  1414 22 0        0 41 42 1 11776 11 41   40220100000343134 42 113600 8112          13 1110000     00
116RW6      1  16  1   1 10039641120141379 6 2 1 6 6 21191 47 10   1  1414 22 1 1003668 41 42 1 01776 30141   30210110100343531 24 113000 8111110   0   13 1100000     00
117RW6      1  17  1   1 10039641120141379 7 2 0 6 7 11141 46 10   1  1414 22 0        0 41 42 1 11776 21 13    6131110000343534 45 113910 7111000   1   13 1100000     00
```

Apart from this data file we also need data about the geological information about the survey clusters of Rwanda. For that we downloaded a shape file named RWGE72FL in zip format. We used an online Google Earth Pro tool to open this file and converted it into a csv file format. Sample of the raw data was presented in figure 8. This file contains a total of 492 rows indicating 492 DHS clusters and 22 columns which relate to cluster geological information like longitude, latitude, DHS cluster ID, year, etc. But we will use only a few columns in that file.

Figure 9

File Containing Cluster Locations

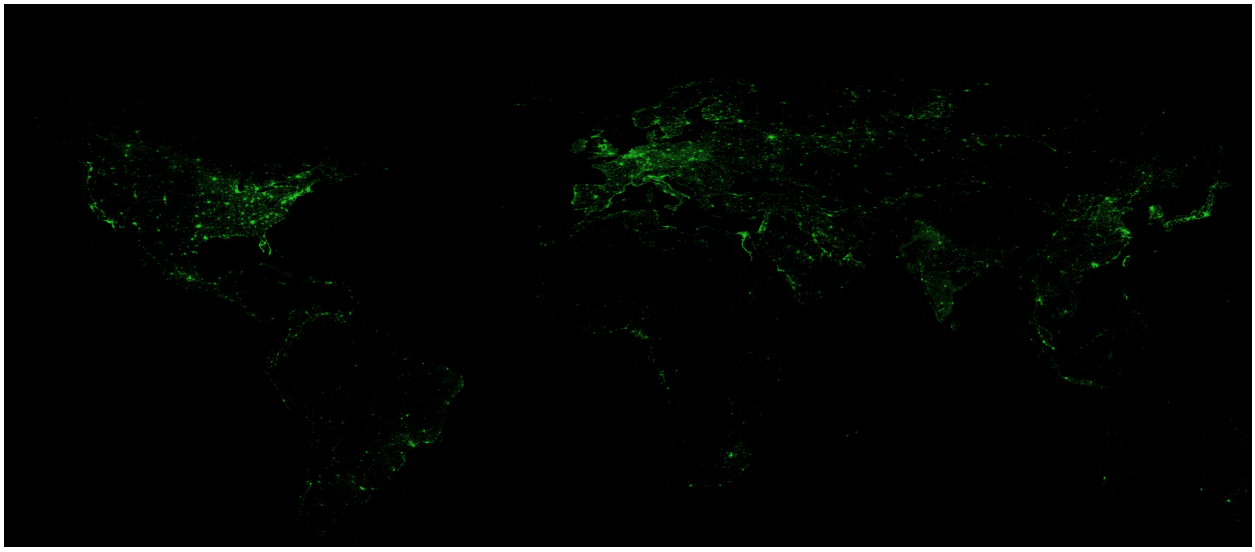
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	DHSID	DHSCC	DHSYEAR	DHSCLUST	CCFIPS	ADM1FIPS	ADM1FIPS	ADM1SALE	ADM1SALE	ADM1DHS	ADM1NAM	DHSREGCC	DHSREGNA	SOURCE	URBAN_RI	LATNUM	LONGNUM	ALT_GPS	ALT_DEM	DATUM
29.71279	-2.50593	RW20140(	RW	2014	1	RW	RW15	Southern	NULL	NULL	2	South	24	Huye	CEN	R	-2.50593	29.71279	9999	1776	WGS84
30.31051	-1.84467	RW20140(	RW	2014	2	RW	RW11	Eastern	NULL	NULL	5	East	53	Gatsibo	CEN	R	-1.84467	30.31051	9999	1651	WGS84
29.57476	-1.84194	RW20140(	RW	2014	3	RW	RW14	Western	NULL	NULL	3	West	35	Ngororero	CEN	R	-1.84194	29.57476	9999	2105	WGS84
30.46364	-2.34816	RW20140(	RW	2014	4	RW	RW11	Eastern	NULL	NULL	5	East	55	Kirehe	CEN	R	-2.34816	30.46364	9999	1414	WGS84
30.16184	-2.2113	RW20140(	RW	2014	5	RW	RW11	Eastern	NULL	NULL	5	East	57	Bugesera	CEN	R	-2.2113	30.16184	9999	1451	WGS84
30.10441	-2.02381	RW20140(	RW	2014	6	RW	RW12	Kigali City	NULL	NULL	1	Kigali City	13	Kicukiro	CEN	U	-2.02381	30.10441	9999	1520	WGS84
29.84323	-1.69171	RW20140(	RW	2014	7	RW	RW13	Northern	NULL	NULL	4	North	42	Gakenke	CEN	R	-1.69171	29.84323	9999	1815	WGS84
30.04251	-1.79965	RW20140(	RW	2014	8	RW	RW13	Northern	NULL	NULL	4	North	41	Rulindo	CEN	R	-1.79965	30.04251	9999	1967	WGS84
30.12038	-1.98773	RW20140(	RW	2014	9	RW	RW12	Kigali City	NULL	NULL	1	Kigali City	13	Kicukiro	CEN	U	-1.98773	30.12038	9999	1400	WGS84
30.44565	-1.96175	RW20140(	RW	2014	10	RW	RW11	Eastern	NULL	NULL	5	East	51	Rwamagar	CEN	U	-1.96175	30.44565	9999	1465	WGS84
30.48936	-1.91733	RW20140(	RW	2014	11	RW	RW11	Eastern	NULL	NULL	5	East	54	Kayonza	CEN	R	-1.91733	30.48936	9999	1543	WGS84
29.7649	-2.20883	RW20140(	RW	2014	12	RW	RW15	Southern	NULL	NULL	2	South	26	Ruhango	CEN	R	-2.20883	29.7649	9999	1794	WGS84
29.42448	-2.26423	RW20140(	RW	2014	13	RW	RW14	Western	NULL	NULL	3	West	31	Karongi	CEN	R	-2.26423	29.42448	9999	2065	WGS84
29.59713	-2.05145	RW20140(	RW	2014	14	RW	RW14	Western	NULL	NULL	3	West	35	Ngororero	CEN	R	-2.05145	29.59713	9999	1698	WGS84
29.62843	-1.84497	RW20140(	RW	2014	15	RW	RW14	Western	NULL	NULL	3	West	35	Ngororero	CEN	U	-1.84497	29.62843	9999	1750	WGS84
30.77163	-2.10467	RW20140(	RW	2014	16	RW	RW11	Eastern	NULL	NULL	5	East	55	Kirehe	CEN	R	-2.10467	30.77163	9999	1295	WGS84
29.86599	-2.10871	RW20140(	RW	2014	17	RW	RW15	Southern	NULL	NULL	2	South	26	Ruhango	CEN	R	-2.10871	29.86599	9999	1558	WGS84
29.46505	-1.63518	RW20140(	RW	2014	18	RW	RW14	Western	NULL	NULL	3	West	34	Nyabihu	CEN	U	-1.63518	29.46505	9999	2377	WGS84
29.98134	-2.25784	RW20140(	RW	2014	19	RW	RW11	Eastern	NULL	NULL	5	East	57	Bugesera	CEN	R	-2.25784	29.98134	9999	1423	WGS84
29.67955	-2.33315	RW20140(	RW	2014	20	RW	RW15	Southern	NULL	NULL	2	South	21	Nyanza	CEN	R	-2.33315	29.67955	9999	1637	WGS84
29.46917	-2.16701	RW20140(	RW	2014	21	RW	RW14	Western	NULL	NULL	3	West	31	Karongi	CEN	R	-2.16701	29.46917	9999	2106	WGS84
29.80958	-2.32621	RW20140(	RW	2014	22	RW	RW15	Southern	NULL	NULL	2	South	21	Nyanza	CEN	R	-2.32621	29.80958	9999	1670	WGS84
30.06735	-2.01624	RW20140(	RW	2014	23	RW	RW12	Kigali City	NULL	NULL	1	Kigali City	13	Kicukiro	CEN	R	-2.01624	30.06735	9999	1645	WGS84
29.30537	-1.61788	RW20140(	RW	2014	24	RW	RW14	Western	NULL	NULL	3	West	33	Rubavu	CEN	R	-1.61788	29.30537	9999	2042	WGS84
29.8398	-1.40526	RW20140(	RW	2014	25	RW	RW13	Northern	NULL	NULL	4	North	44	Burera	CEN	U	-1.40526	29.8398	9999	2115	WGS84
30.44411	-1.43481	RW20140(	RW	2014	26	RW	RW11	Eastern	NULL	NULL	5	East	52	Nyagatare	CEN	R	-1.43481	30.44411	9999	1475	WGS84
29.28951	-1.7191	RW20140(	RW	2014	27	RW	RW14	Western	NULL	NULL	3	West	33	Rubavu	CEN	R	-1.7191	29.28951	9999	1673	WGS84

Night Time Satellite Images

For nighttime satellite images our group finalized on NOAA over Earthdata, this is due to Earthdata not containing high pixelated night time images. Images from NOAA are also tagged with geological information and light intensity. The light intensity was shown as a number between 0 and 63. The files offered are cloud-free composites derived from all archival DMSP-OLS smooth resolution data for calendar years. The grids are 30 arc seconds long and cover a latitude range of -65 to 75 degrees (Earth Observation Group - Defense Meteorological Satellite Program, Boulder, n.d.). The location used for this project is Rwanda in East Africa. World wide night time images are available from the year 1992 to 2013 on the website. We are utilizing the most recent accessible image, which is from 2013. The downloaded image is 726 MB in size and has a high resolution, which keeps some quality in the images collected for a specific area like Rwanda. The image acquired from the website is shown in figure 10 below.

Figure 10

Image Representing Nightlight Data



Daytime Satellite Images

We discovered that the images lacked quantity after extensively reviewing all dataset sources, including the National Weather Service. For daytime we needed more quantity and found out that Google Static Maps contained the required amount. However, we needed to obtain an API key to access the images. So, we created a Google Cloud account and created a project and purchased an API key to access it, and will retrieve daytime images using that API key. Now we need to configure those images so that they may be downloaded from Google. We need parameters like zoom level, coordinates of latitude and longitude, size in terms of pixels to download the image. That is why we require another file with information on the borders of Rwanda's DHS clusters. That file was obtained from the National Spatial Data Infrastructure (NSDI) website (Data Page, n.d.). Since the download file is in the shape file format, it is difficult to access using standard applications such as notepad and excel. To access them, we

need specialized software such as ArcGIS. This file can be opened using another online tool, called Google Earth Pro. Now we have details of geological locations to download the images.

Now we have finalized details of geological locations to download the images. We also decided on the image size of 400 X 400 and the zoom level of 16 for the images to be downloaded. We first gathered the images using information from the NSDI file, and then we grabbed the light intensity information from that cluster from the night time image and saved it into that file. That is, all the images collected from Google are arranged in light intensity order. As a result, all of the images are now tagged with light intensity. Figure 11 shows a sample of the image collected from the Google map static API required for the project.

Figure 11

Image Representing Daytime Images

Data Pre-Processing

Data preprocessing is a vital stage that includes handling all types of issues in raw data and tackling them in an efficient way to fix them. We now have two types of datasets. The first is

an image dataset, while the second is a survey dataset. The steps involved in data pre-processing for such datasets are shown below.

Survey Data

The survey dataset was obtained from the DHS website. The US Agency for International Development conducts these surveys (USAID). The survey data file contains all of the information regarding the survey, as well as Rwanda cluster information created from the Rwanda shape file. We opened this file in a spreadsheet which showcased 12,699 rows and hundreds of columns. When we first imported the dataset, we conducted some preliminary data exploration and discovered that there are no missing values. To check for noise on our survey data, we did some preliminary text pre-processing to check for inconsistencies. There were over a hundred columns, and some of them had white spaces, which we removed using the str.strip() method. We also found a few outliers in our survey dataset by applying a normal distribution; data points that were below mean-3*(sigma) or above mean+3*(sigma) were designated outliers. To deal with these outliers, we applied the InterQuartile Range (IQR) proximity rule. Outliers are data points that fall below Q1 - 1.5 IQR or above Q3 + 1.5 IQR and were eliminated. Label encoding was also used to convert categorical labels into numeric form so they would be in a machine readable format.

The dataset comprises all of the data from household surveys. The data in this file is in text, integer, and float formats. The important thing to remember here is that we do not want all of the columns from this file. We are only interested in a few columns that have to do with cluster information and the wealth index. To begin, we will filter the columns required for our project. The file had two columns entitled V001, HV270 in the file representing Cluster ID and wealth index, respectively. Data validation is an essential part in the pre-processing process since

it ensures the quality, amount, and reliability of the data for our project. As part of the data validation in our project, we will analyze the range of the wealth index and the geological information of the clusters. Figure 12 showcases the range of the wealth holding index from the original dataset. The wealth holding index ranges from -99,877 to 396,455, which is a massive amount to handle. As a result, we must apply an appropriate method to express these numbers in a smaller range.

Figure 12

Cluster with raw wealth index range

	cluster	wlthindf
count	12699.000000	12699.000000
mean	246.536971	1805.954012
std	141.974045	93372.953854
min	1.000000	-99877.000000
25%	124.000000	-62794.500000
50%	247.000000	-31859.000000
75%	370.000000	28097.000000
max	492.000000	396455.000000

As we are gathering information from households across the country, we have a large number of rows for a single cluster. However, each cluster requires only one value of the wealth index. There are several approaches that may be used. Some of them compute the cluster mean and use it as a final wealth index for the cluster. However, there is a disadvantage with this approach i.e., outliers will affect the mean value which leads us to lose quality in the data. To prevent this, we will handle this issue by using the median value. The median value is unaffected by outliers and also simple to compute. As a result, we aggregated all of the data by cluster ID,

then took the median from each group and gave that value as the final wealth index value to the corresponding cluster.

Night Light Images

The gathered night light image of the world is downloaded in tar file. This tar file comprises eight files. These are the raw average visible band files, cleaned up average visible band files, cloud free coverage data files, and a readme file. We will only use one of these files, namely the cloud free coverage data file. The readme file helped us understand the characteristics and situations under which the image was captured. As a result, the image was clear of clouds and other flaws. Aside from that, these image files are in tiff format, which stands for tagged image format. As a result, the image will have all of the geological information in an easy to manage format.

Day Time Images

We obtained 24,877 images from Google Static Maps. These images have a resolution of 400 X 400 and a zoom level of 16. These images are cloud-free since they were downloaded from a Google Static Map. As a result, these images don't have much room for pre-processing. Some images, however, are downloaded without any meaning, such as having the same color filled in the majority of the image. The images in doubt were removed from the dataset. We need labels for the downloaded images, for that we considered the light luminosity of the downloaded images and labeled them with the number, which ranged from 0 to 64.

Data Transformation

Data transformation is the process of transforming datasets into the necessary formats and values for model input. There are three types of datasets. We want to use machine learning and

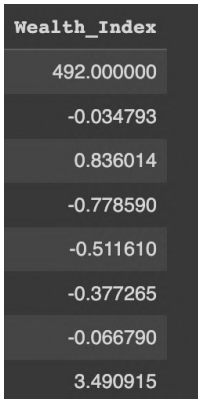
deep learning models on these datasets for this project. As a result, we must change the data into the following formats.

Survey Data

Now that we have clean preprocessed survey data we can proceed with the workflow before applying the model however we need to modify the data to make it suitable for our approach. As discussed in the previous section, we must change the wealth index, which has a wide range. We divide each wealth index value from the data file by one lakh, so that each wealth index changes to a float number ranging from -0.34 to 3.49. Figure 13 illustrates the wealth index contains the transformed range of the wealth index in the dataset.

Figure 13

Transformed wealth index



Wealth_Index
492.000000
-0.034793
0.836014
-0.778590
-0.511610
-0.377265
-0.066790
3.490915

Normalizing data before training machine learning models is also a common procedure. We will normalize our Survey Data in order for our Machine Learning algorithms to function better. Normalization also makes the training process less sensitive to feature size. This results in getting better coefficients after training. We will utilize the MinMaxScaler() function from the sklearn package to do normalization. We have now generated a new file with the information of the cluster with the wealth index, we named that file as cluster with wealth index.csv file. Figure

14 below shows the cluster ID with aggregated data, after which we picked the median from each group and assigned that value to the appropriate cluster as the final wealth index value.

Figure 14

Normalized wealth index of survey data

```
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(data_all)

[35] norm_df = pd.DataFrame(normalized_data, columns=data_all.columns)

[36] norm_df.head()
```

	id	max_	min_	mean_	median_	std_	wealth
0	0.000000	0.0	0.0	0.0	0.0	0.0	0.252549
1	0.002037	0.0	0.0	0.0	0.0	0.0	0.078530
2	0.004073	0.0	0.0	0.0	0.0	0.0	0.073972
3	0.006110	0.0	0.0	0.0	0.0	0.0	0.055734
4	0.008147	0.0	0.0	0.0	0.0	0.0	0.133179

Night Time Images

We have a nighttime view of the entire planet with light brightness. Our next objective is to convert the image data into a format appropriate for training machine learning and deep learning models. We must merge the nighttime image with the DHS data and build a new data file. The initial stage in this process is to detect the clusters in the night light image using information from the newly created data file which contains latitude and longitude. Next, for each cluster, we will examine the light intensity of each pixel and calculate the average, minimum, maximum, median, and standard deviation of light intensity. We will then create a new file that includes all of this information. As a result, the newly generated file will have 492 rows and six columns, including the cluster ID. This new file will be used to train machine learning models. Figure 15 shows the night time image features that were extracted from the night time satellite image. These attributes are associated with the wealth holding index.

Figure 15

Extracted features from night time image along with wealth index

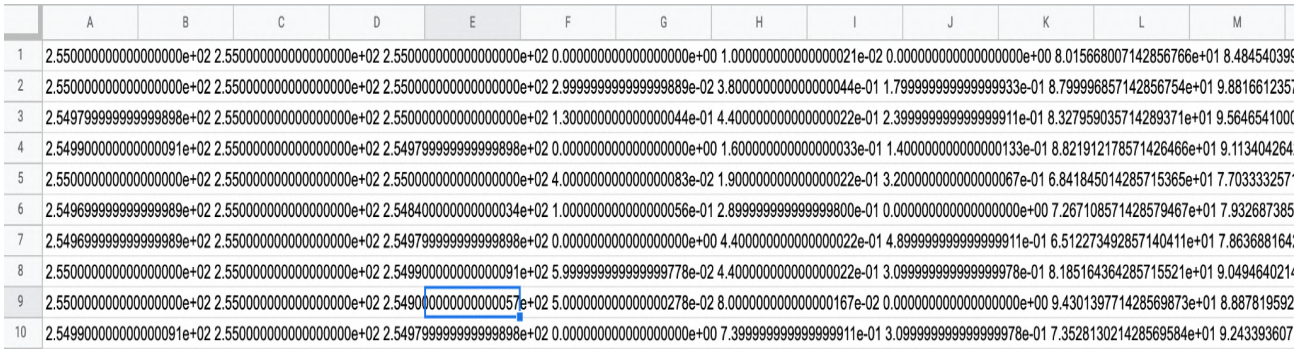
	id	max_	min_	mean_	median_	std_	wealth
2	1	0	0	0	0	0	0.29967
3	2	0	0	0	0	0	-0.443305
4	3	0	0	0	0	0	-0.462765
5	4	0	0	0	0	0	-0.540635
6	5	0	0	0	0	0	-0.20998
7	6	49	0	12.42	8	11.56043252	1.667235
8	7	0	0	0	0	0	-0.38013
9	8	6	0	0.12	0	0.84	0.040095
10	9	63	6	33.43	28	22.07272299	2.83783

Day Time Images

Now that we have all of Rwanda's images from Google's Static Maps. Unfortunately, we cannot utilize such images directly to train machine learning models. So, now we'll extract image features from the images and generate a file. As part of it, we evaluate each image and extract the pixel band values for the image. After creating a list of the image's band values, we extract the maximum, minimum, mean, median, and standard deviation band values, as we previously did with nighttime images. Now, we extract features from the image and make a file with 16 columns including the wealth index. Figure 16 shows the Day time image features that were extracted from the Google Static Maps. These features are also correlated with the wealth holding index.

Figure 16

Extracted features from day time images along with wealth index



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	2.550000000000000000e+02	2.550000000000000000e+02	2.550000000000000000e+02	0.000000000000000000e+00	1.000000000000000021e-02	0.000000000000000000e+00	8.015668007142856766e+01	8.484540399					
2	2.550000000000000000e+02	2.550000000000000000e+02	2.550000000000000000e+02	2.999999999999999889e-02	3.800000000000000044e-01	1.799999999999999933e-01	8.799996857142856754e+01	9.881661235					
3	2.549799999999999898e+02	2.550000000000000000e+02	2.550000000000000000e+02	1.300000000000000044e-01	4.400000000000000022e-01	2.399999999999999911e-01	8.327959035714289371e+01	9.564654100					
4	2.549900000000000091e+02	2.550000000000000000e+02	2.549799999999999898e+02	0.000000000000000000e+00	1.600000000000000033e-01	1.400000000000000133e-01	8.821912178571426466e+01	9.113404264					
5	2.550000000000000000e+02	2.550000000000000000e+02	2.550000000000000000e+02	4.000000000000000083e-02	1.900000000000000022e-01	3.200000000000000067e-01	6.841845014285715365e+01	7.703333257					
6	2.549699999999999989e+02	2.550000000000000000e+02	2.548400000000000034e+02	1.000000000000000056e-01	2.899999999999999800e-01	0.000000000000000000e+00	7.267108571428579467e+01	7.932687385					
7	2.549699999999999989e+02	2.550000000000000000e+02	2.549799999999999898e+02	0.000000000000000000e+00	4.400000000000000022e-01	4.899999999999999911e-01	6.512273492857140411e+01	7.863688164					
8	2.550000000000000000e+02	2.550000000000000000e+02	2.549900000000000091e+02	5.999999999999999778e-02	4.400000000000000022e-01	3.099999999999999978e-01	8.185164364285715521e+01	9.049464021					
9	2.550000000000000000e+02	2.550000000000000000e+02	2.549000000000000057e+02	5.000000000000000278e-02	8.000000000000000167e-02	0.000000000000000000e+00	9.430139771428569873e+01	8.887819592					
10	2.549900000000000091e+02	2.550000000000000000e+02	2.549799999999999898e+02	0.000000000000000000e+00	7.399999999999999911e-01	3.099999999999999978e-01	7.352813021428569584e+01	9.243393607					

Data Preparation

Having done the necessary pre-processing on the datasets, we have three files to deal with. The first data file is an integrated dataset of DHS and night time image data, while the second is a Google image dataset with DHS data. The first two files contain 492 rows, one for each cluster in Rwanda, as well as 6 columns in the first file and 16 columns in the second, which includes the wealth index. The third dataset comprises a total of 24,877 images retrieved from Google Maps.

First, we separated the image dataset into three subsets, training, validation, and testing, using a 70-10-20 split respectively as given in Figure 17, 18 and 19.. As a result, the training dataset contains 17,414 images, the validation dataset contains 2,488 images, and the testing dataset contains 4,975 images. We will apply models to the training image dataset and use the extracted features from the images with the DHS daytime dataset to get results. We will add another stage of transfer learning and will use the night time DHS data to restrain the model for better performance. To ensure effective training, we will use the validation dataset while training the model and alter the hyper parameters of the training model as needed. Following the completion of the model's training, we will utilize the testing dataset to test and assess the

model's performance in a variety of efficient methods. Then we will compare the models' performance and proceed with the best performing model.

Figure 17

Sample of Training Set

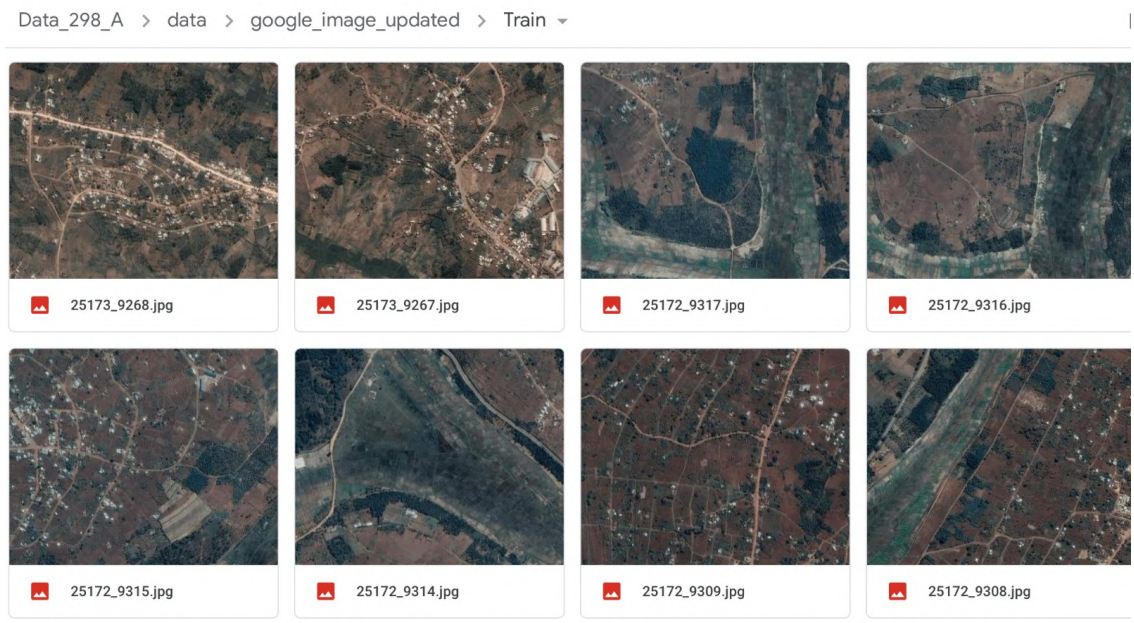


Figure 18

Sample of Testing Set

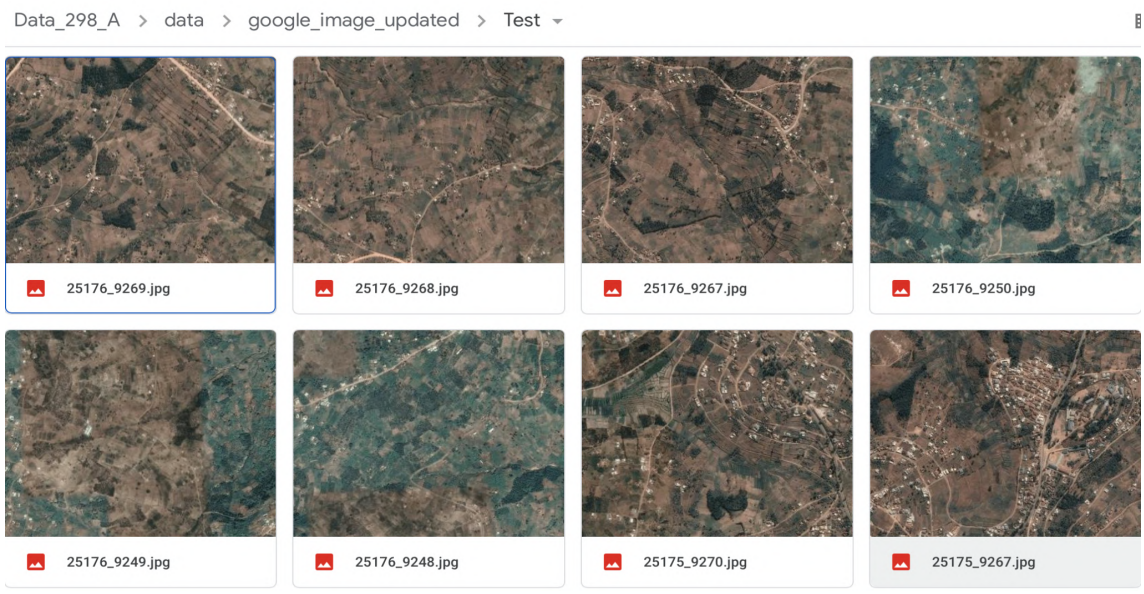
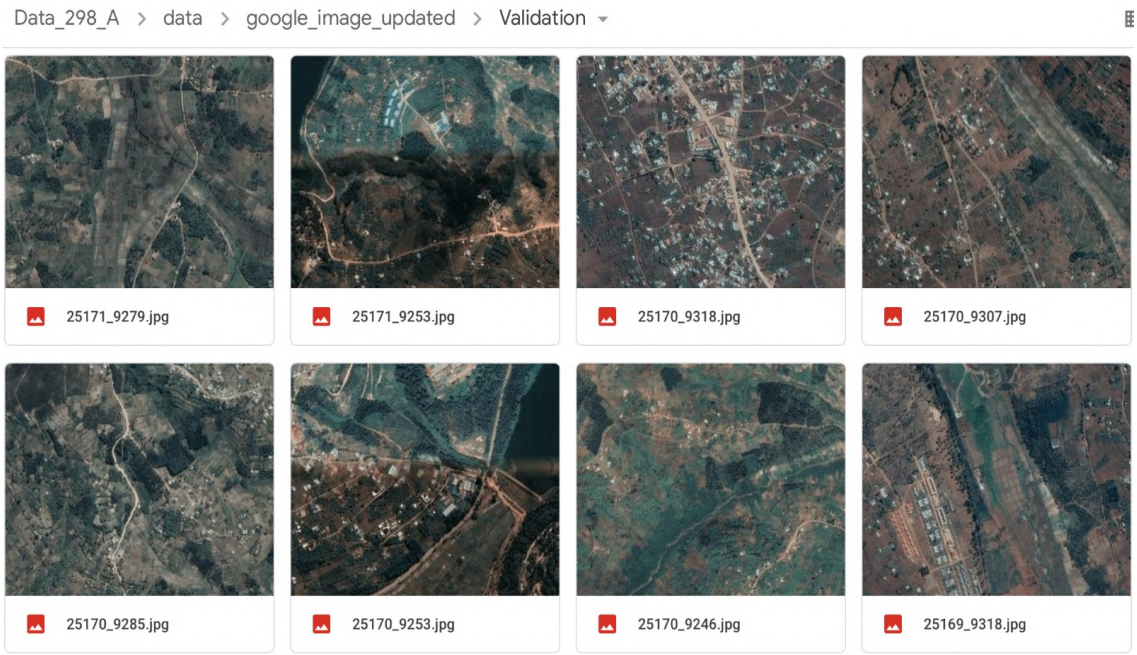


Figure 19

Sample of Validation Set



Data Statistics

We have pre-processed the dataset of images and the data file of daytime and nighttime with DHS survey in this stage. We ran a few statistical procedures on the provided datasets and displayed them using Python libraries. First, we will do statistical analysis on DHS combined data with nighttime images. The figure 20 below depicts the dataset, which has 492 rows and 6 columns. All of the columns consist of float data type, and the figure 21 depicts the basic statistical analysis on the dataset in column order. The figure shows the mean, standard deviation, minimum, maximum, and quartiles. We can observe from the graph that the min_i values are nearly identical, and the standard deviation of that column is far lower than that of the other fields.

Figure 20

Description of Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 492 entries, 0 to 491
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Cluster ID    492 non-null    float64
 1   Max_l         492 non-null    float64
 2   Min_l         492 non-null    float64
 3   Mean_l        492 non-null    float64
 4   Median_l      492 non-null    float64
 5   std_l         492 non-null    float64
 6   Wealth_Index  492 non-null    float64
dtypes: float64(7)
memory usage: 27.0 KB
```

Figure 21

Statistics of Each Column in Data

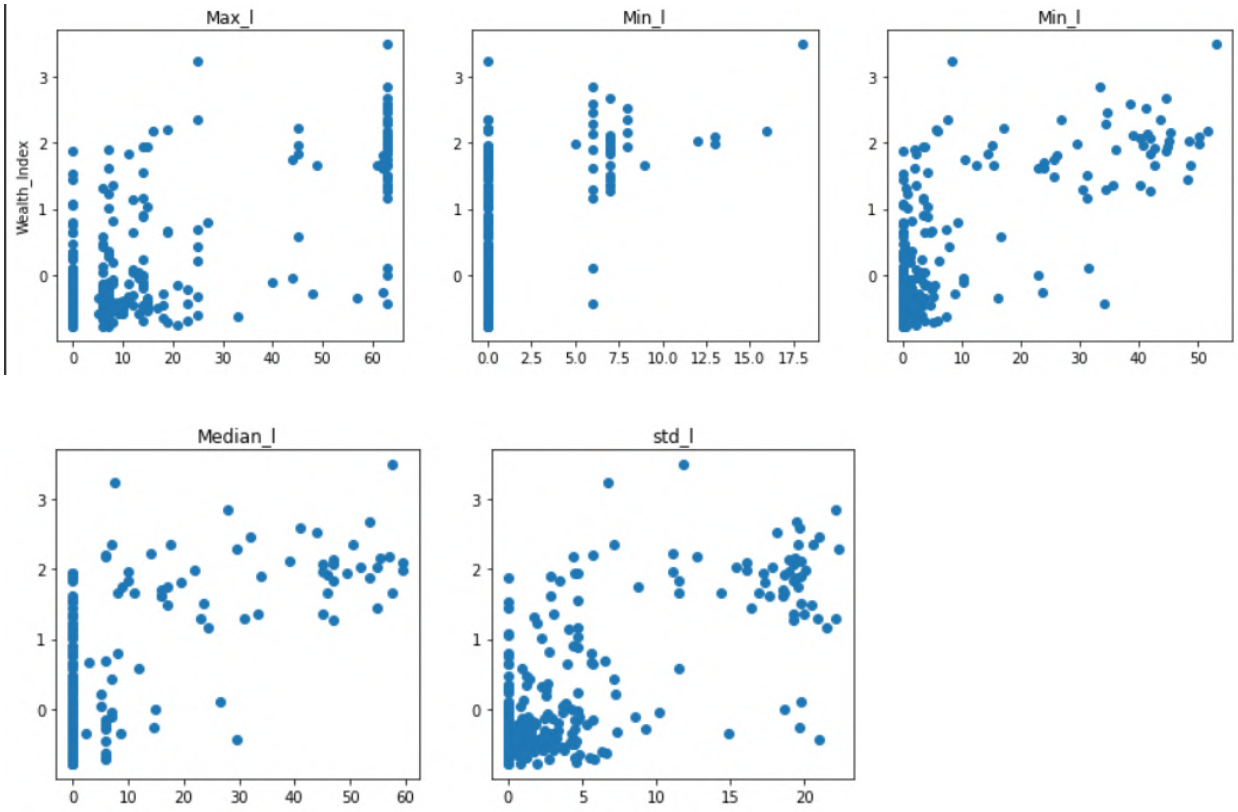
	Cluster ID	Max_l	Min_l	Mean_l	Median_l	std_l	Wealth_Index
count	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000
mean	246.500000	10.199187	0.587398	4.392602	3.946138	2.875071	-0.034793
std	142.172431	19.070054	2.204070	11.248362	11.928489	5.636129	0.836014
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.778590
25%	123.750000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.511610
50%	246.500000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.377265
75%	369.250000	9.000000	0.000000	1.430000	0.000000	2.847446	-0.066790
max	492.000000	63.000000	18.000000	53.190000	59.500000	22.253036	3.490915

As shown in figure 22, we produced scatter graphs between all of the columns and the wealth index. The graph allows us to see the patterns of these columns with their associated

wealth indexes, which allows us to draw certain conclusions. When we observe from the graphs, as Max_i and std_i increases the associated wealth index is also increasing. This can be a valuable graph conclusion.

Figure 22

Scatterplots of Each Column Versus Wealth Index

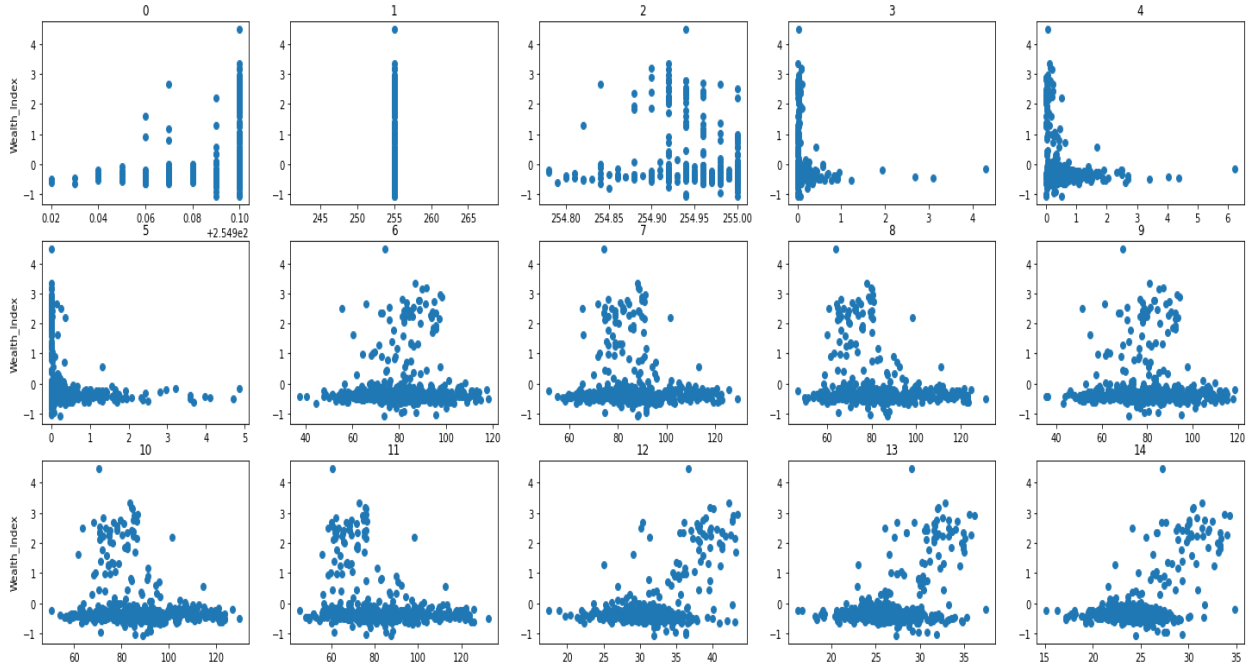


Now, we considered a data file of day time images with DHS survey data. This file has 492 rows and 16 columns. All columns indicate the characteristics extracted from the image. The columns all have the float data type. As shown in Figure 23, we created many scatter plots between all the columns with the wealth index. This demonstrates the relationship between the column and the wealth index. As observed in the graph, certain columns, like columns 5, 6, 10, and so on, exhibit a positive relationship with the wealth index. On the other hand, column 1

does not show such a relationship. Additionally, it is worth noting that nearly all rows in the dataset have the same value in column 1..

Figure 23

Daylight Image Features Versus Wealth Index



These are the few statistics done on the data as shown in the above visualizations. Now we will compare the dataset from pre-process and after pre-process. Before preprocessing the dataset, the original survey file collected from DHS website consists of 12,699 rows with hundreds of columns. But identifying the clusters and wealth index in that file, calculated the average wealth index and transformed the original wealth index to a ratio then the new file consists of only 492 rows as shown in the figure 24 and 4 columns i.e., cluster ID, longitude, latitude and transformed wealth index.

Figure 24

```
[15] len(df.cluster.unique())

     492
```

Before pre-processing the image we have 25,789 number of images downloaded from google static map api. After doing the preprocessing on the images like eliminating the blur images and images that are covered with clouds all over we have 24,877 number of images as shown in the figure 25.

Figure 25

```
['0', '5', '6', '7', '12', '17', '22', '29', '30', '26', '16', '11',
File count: 24877
```

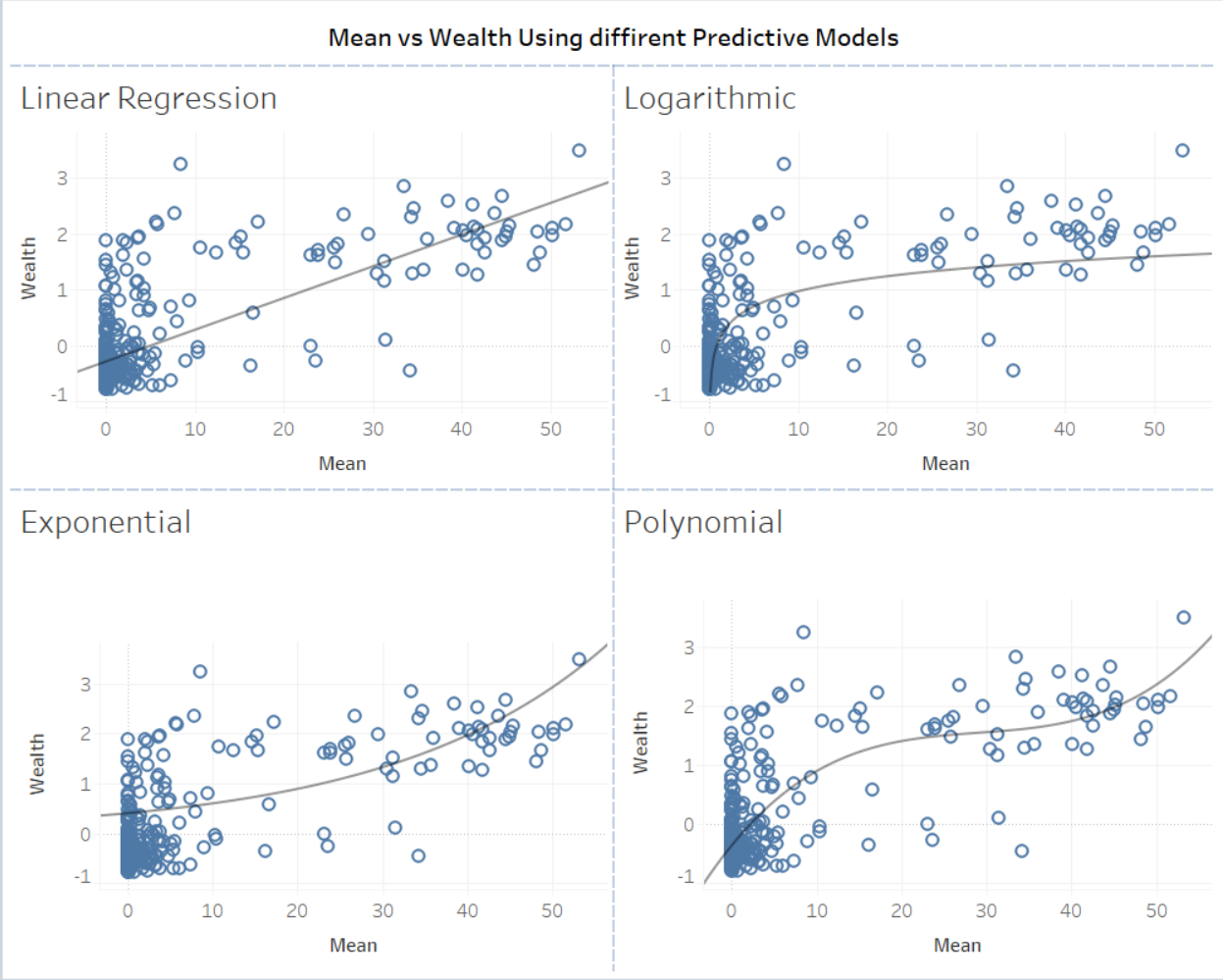
Data Analytics Results

As seen in the above steps, raw data passed through many processing stages before becoming ready for model building. We reasoned that we needed a more capable tool than Python to evaluate the prepared data. Tableau is one such tool that is more widely used in the data analytics industry. We utilized this tool to examine the DHS with nighttime image data as well as the DHS with daytime image data. To assess excellent relationships between the columns and the desired wealth index, we need a decent model. Tableau has a prediction model option. We used that and created graphs showing the relationships between the columns and the wealth index. Each graph depicts the wealth index as it relates to a certain column. We tested all of the columns and provided only a handful that made sense in terms of understanding the relationship. As illustrated in Figure 26, the wealth index is calculated as a function of the mean_i in column using various models, including linear regression, logarithmic, exponential, and polynomial.

Upon examination, it is evident that the logarithmic and polynomial graphs provide more meaningful insights compared to the other two graphs.

Figure 26

Predictive Models of Mean Column With Wealth Index

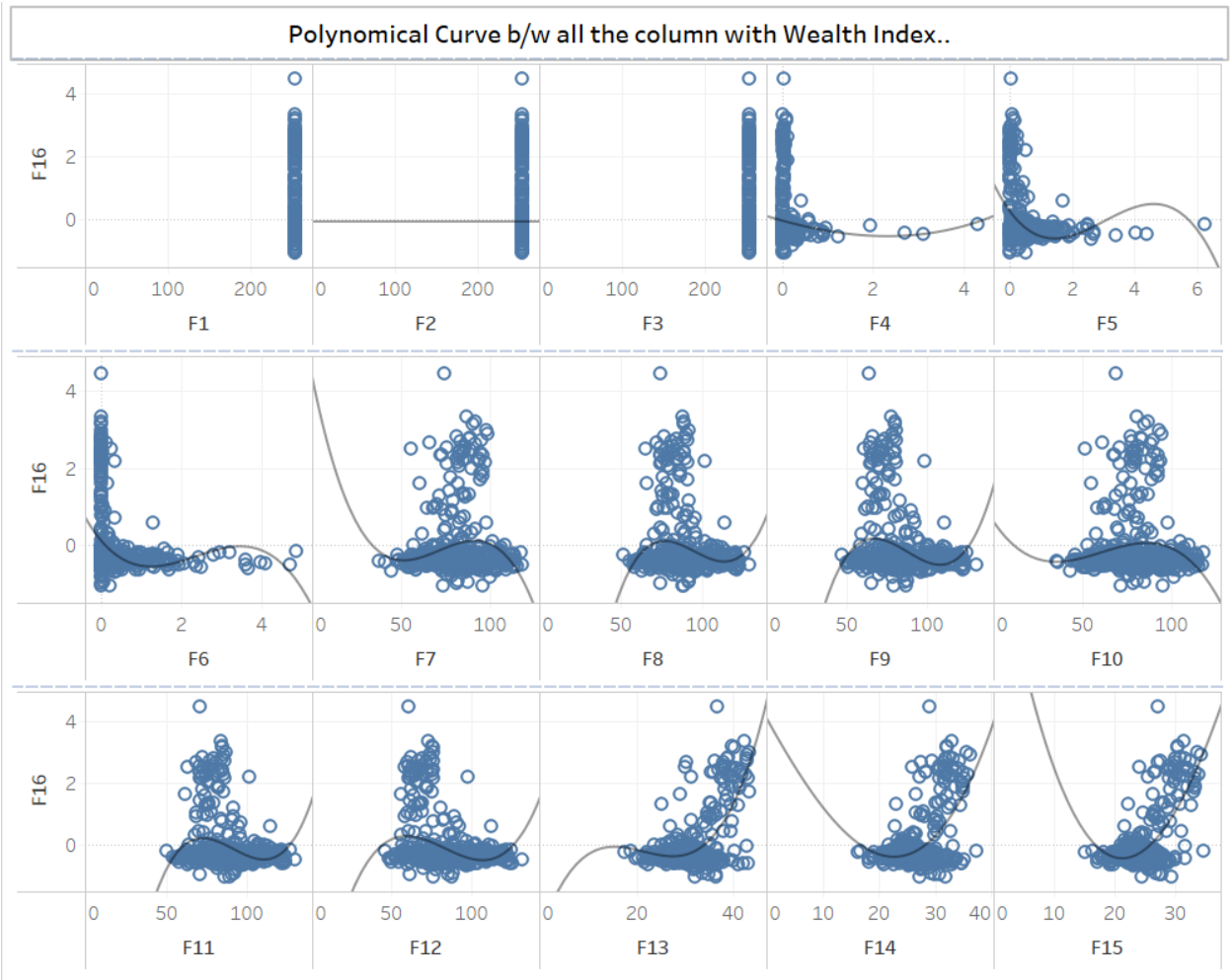


Consider the DHS in the context of a daytime dataset. We have 16 columns, one of which is a wealth index. We used the same method to create the polynomial prediction model and plot graphs connecting all of the columns with the wealth index. As seen in the figure 27, the initial characteristics, F1, F2, and F3 do not produce any relevant graphs or models with the wealth index. However, other variables such as F6, F12, and so on have a substantial association with

the wealth index. We cannot conclude that attributes F1, F2, and F3 are significant to the wealth index. There is no relationship with the wealth index on its own, however, it may be beneficial when combined with the other characteristics. Non-linear relations and complex relations can be built using deep learning. That's why we opted for deep learning in this project to build a model.

Figure 27

Polynomial Curve of All Daytime Image Features With Wealth Index



Model Development

Model Proposals

This section will include the proposed models which we will use to predict the wealth index of a place using the images of that place and also we need some regression models for determining whether features extracted from the images have any relationship with the wealth index. Regression methods including Linear Regression, Ridge Regression, Lasso Regression, and Random Forest regression are employed in this method. For extracting the features from the images we will utilize deep learning models such as VGG Net, ResNet, InceptionNet, DenseNet and also a customized model which will be an enhanced version of the base ResNet model.

For solving the proposed problem of predicting the wealth and night light intensity of the place, we will need regression models as well as deep learning models. The use of regression models will be utilized to see whether the features extracted from the nightlight images and day time images are useful for our problem or not. With these initial steps we will decide the best performing regression model for our problem. Later, we will introduce transfer learning techniques to extract the features from the google daytime images and as well as the night light images, and apply the selected regression model from the previous step to predict the poverty index of the place and the night light intensity.

VGG Net

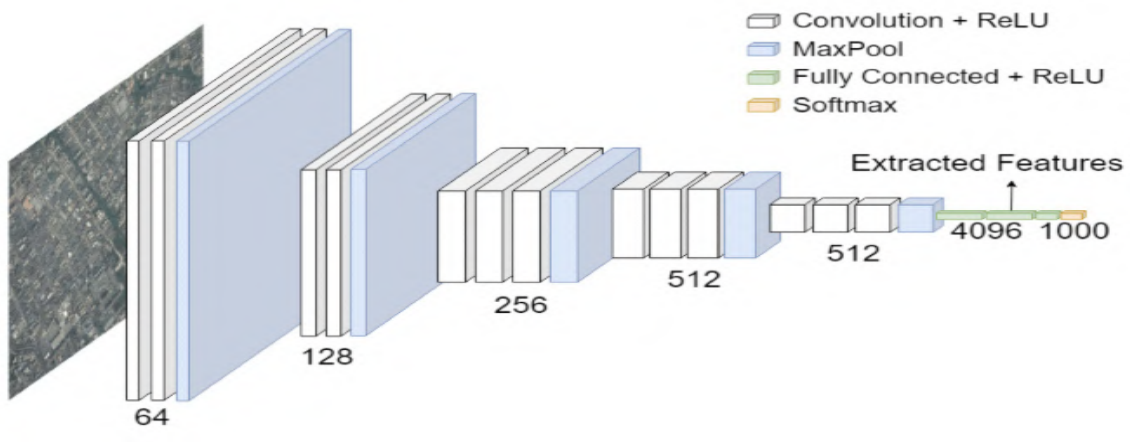
In the 2014 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) competition, Karen Simonyan and Andrew Zisserman developed the Convolutional Neural Network (CNN) model known as VGG16. The ILSVRC is a yearly contest for deep learning systems that identify things in pictures or videos. Authors proposed VGG16 in 2014 in the above mentioned competition which outperformed the other models. VGG16 can classify the images of 1000

different classes with 92.7% accuracy and also has a flexibility of using it with transfer learning (Simonyan & Zisserman, 2014).

VGG 16 consists of 16 layers, 13 of which are convolutional layers and 3 are fully connected layers, including the final output layer. Apart from that this network contains five max pooling layers also, figure 28 shows the architecture of VGG-16. The first layer network is the input layer of 224 X 224 size of RGB image (3 channels). We can divide the network into six blocks in which five blocks are of CNN and one is of dense layer. Each CNN block contains two layers of CNN and one max pooling layer. Every CNN layer in the network is with the same filter size of 3X3 and stride of 2. CNN layers of block one contains 64 filters and the other CNN layers of other blocks contain 128, 256, 512 filters. Last block contains three dense layers with the 4096 neurons in two layers and 1000 neurons in the last layer as having 1000 classes to classify. That's why the last CNN layer is subjective to change depending on the problem.

Figure 28

Architecture of VGG-16



Note. Image of architecture of VGG 16 taken from "*Structure of VGG 16*" by Ye Ni 2021.

https://www.researchgate.net/publication/342909321_An_Investigation_on_Deep_Learning_Approaches_to_Combining_Nighttime_and_Daytime_Satellite Imagery for Poverty

DenseNet

Developed in collaboration between Cornwell University, Tsinghua University, and Facebook AI Research (FAIR), DenseNet is a Convolutional Neural Network (CNN) model that earned the best paper award at the 2017 Computer Vision and Pattern Recognition Conference.

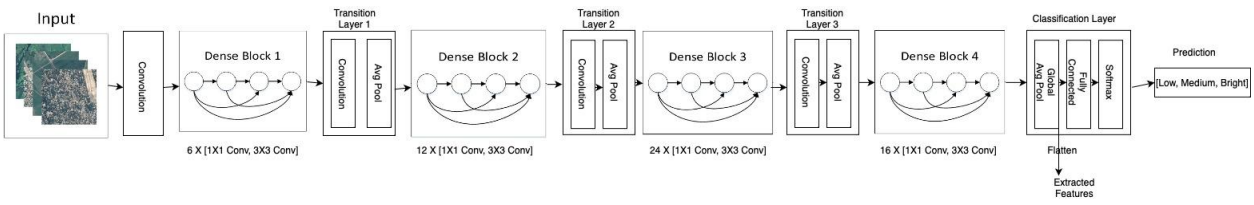
In contrast to ResNet's proposal to deepen the network and Inception-proposal Net's promise to spread the network, DenseNet promises to reuse the multilevel characteristics through dense shortcut connections between layers. Results from the experiment indicate that DenseNet outperforms ResNet.

Figure 29 depicts the DenseNet topology. We can tell that it is made up of a number of dense blocks. Dense shortcut links are planned for each block. Assume that a block contains L layers. There will be (L(L + 1))/2 direct connections between levels. Due to its extensive network of connections, it is known as DenseNet. The formula (1) yields the output of the Lth layer.

$$xl = Hl([x0,x1,...,xl1]) \quad (1)$$

Figure 29

Architecture of DenseNet



Note. Architecture of DenseNet adopted from "*Structure of DenseNet-121*" by Ye Ni. https://ieeexplore.ieee.org/document/9139344

The DenseNet used in the experiments is composed of three dense blocks, each with an equal number of layers. Prior to entering the first dense block, the input images are treated to a convolution with 16 output channels. Between two adjacent dense blocks, a 1x1 convolution and a 2x2 average pooling are employed as transition layers. A softmax classifier is applied once a global average pooling is completed at the conclusion of the final dense block.

We see that although ResNet operates its shortcut connections by additions, DensetNet operates its shortcut connections by concatenation. Therefore, rather than serving as limitations on identity mapping, DensetNet's shortcut connections serve as multiple-level feature reuse. This maximizes information flow and maybe mitigates the identity mapping-related information discarding problem in ResNet. However, the concatenation may result in a considerable increase in the number of feature maps. To solve the issue, the growth rate k, a hyperparameter k, is utilized. In this instance, the 1 x 1 convolution kernels' channel number is k. Fixing the amount of feature maps in each block to k0 + k(L-1) solves the issue, where k0 is the quantity of features created by the previous dense block (Huang et al., 2021).

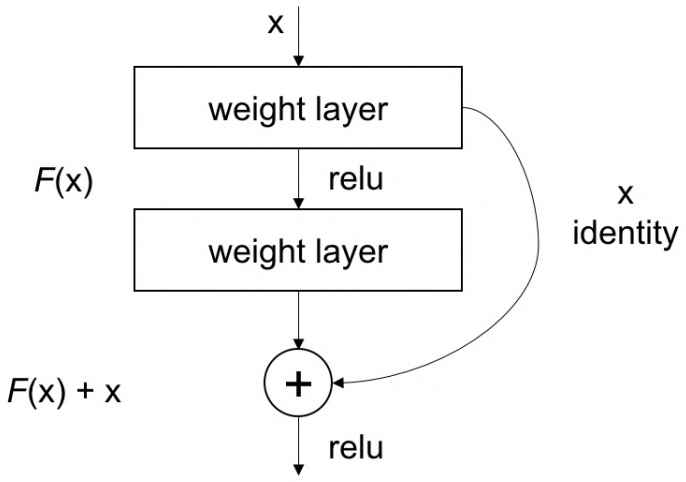
ResNet

The ResNet or Deep Residual learning model was proposed in the year 2015 and won the ImageNet competition in the same year. ResNet is also known as identity shortcut connection which lets the model skip multiple layers and allows intensive batch normalization. After AlexNet won the LSVRC 2012 classification contest, ResNet became the booming one and made a major change in the deep learning community as it can train more than 1000 layers at a time thereby maintaining the compelling performance. One of the major problems that ResNet solves

is vanishing Gradient. Deeper networks may be taught, as demonstrated by this technique ResNet. The accuracy gets more saturated as the network depth increases. It's not even because there are too many parameters or overfitting, but rather because the training error has decreased. The inability to backpropagate the gradients is the cause of this. This may be avoided by sending the gradients with a residual block directly to the deeper layers as shown in figure 30 (Kaiming et al., 2021).

Figure 30

Building Block of Residual Learning



Note. Building Block of Residual Learning adopted from "*Residual Learning: a building block*" by Kaiming He. https://arxiv.org/pdf/1512.03385.pdf

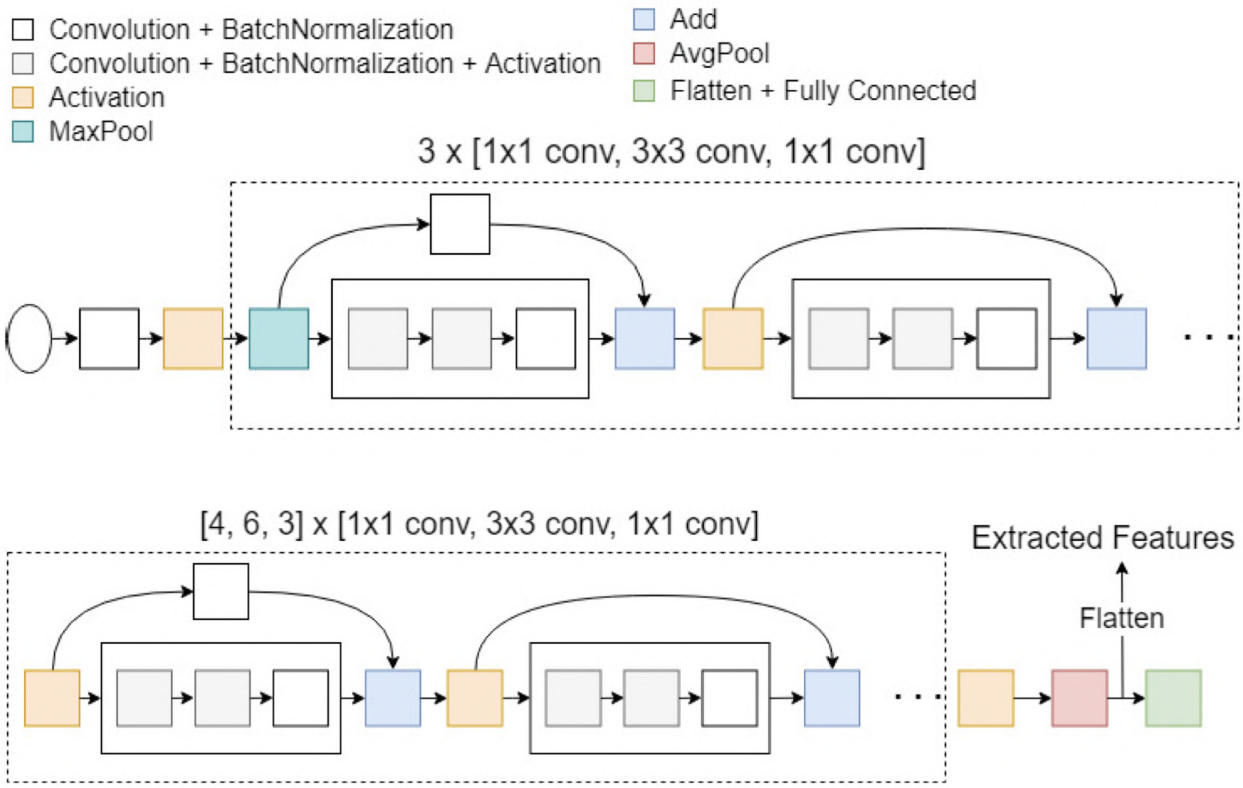
When we figure out how to change the input feature map for better features, this residual function may be seen as a refining stage. A simple network, on the other hand, anticipates that each layer will acquire brand-new feature maps. If no refinement is necessary, the intermediate layers can learn to gradually decrease their weights toward zero, producing a residual block that duplicates an identity function.

Each colored layer of a layer block represents a set of convolutions in the same dimension. Strided convolution is used to routinely downscale the feature mapping while also increasing the channel depth in order to preserve the temporal complexity per layer. When the input is projected using a 1x1 convolution to suit the size of the resultant block, dotted lines stand in for residual connections.

The ResNet 50 architecture is represented in figure 31. In order to lessen the computational load while computing the 3x3 convolution, we simply substitute each two-layer residual block in the ResNet 50 model with a three-layer bottleneck block that utilizes 1x1 convolutions to first lower the channel depth and then restore it.

Figure 31

Architecture of Residual Network.



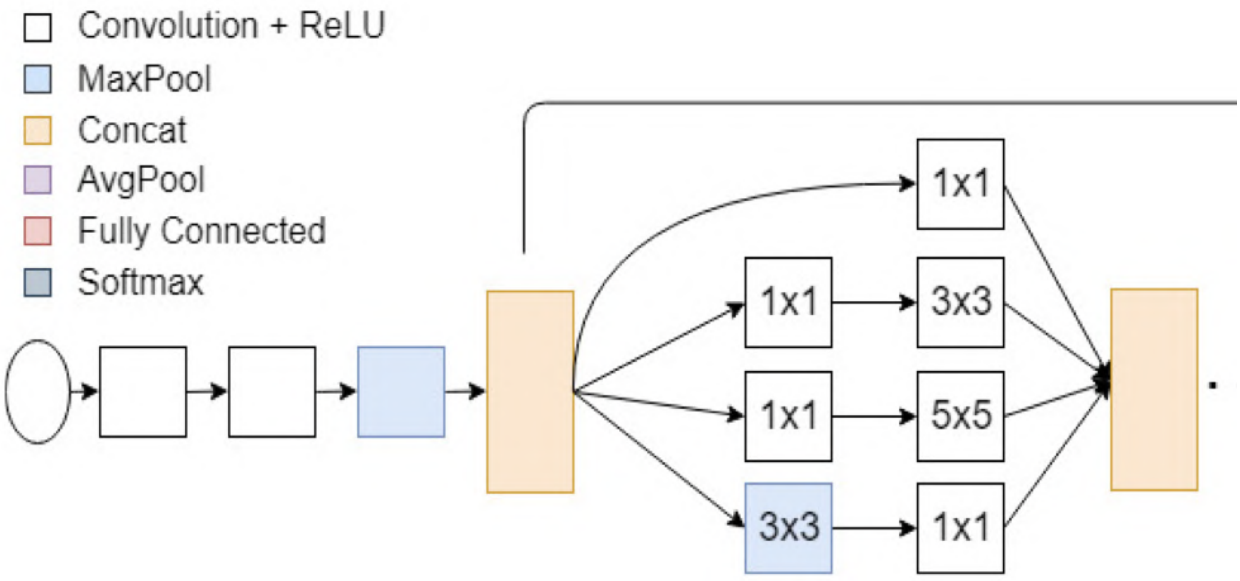
Note. Architecture of ResNet adopted from "*Structure of ResNet-50*" by Ye Ni. https://ieeexplore.ieee.org/document/9139344

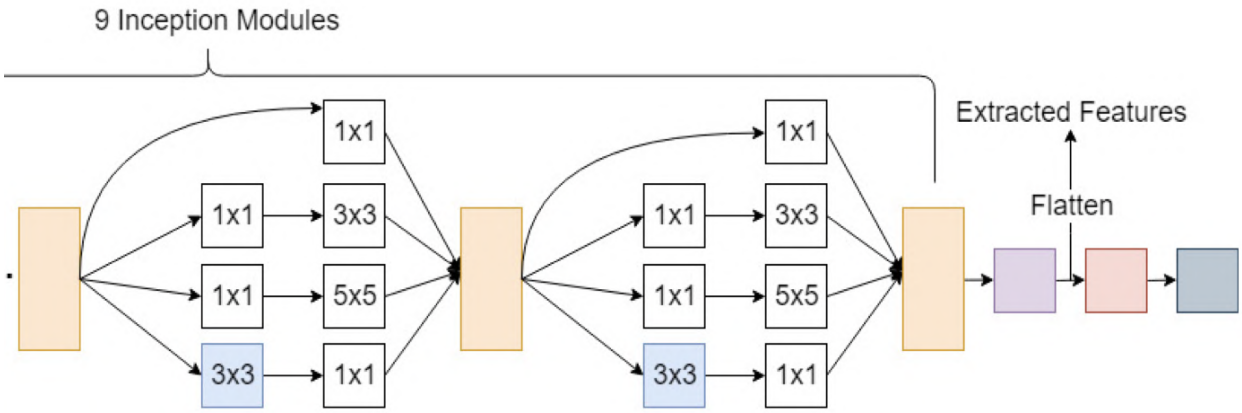
Inception Net

In the 2014 ImageNet Large-Scale Visual Recognition Challenge, Inception Net, a deep convolutional network, achieves a new standard for classification and detection (ILSVRC14). Previously, the Inception network was regarded as a cutting-edge deep learning architecture (or model) for resolving picture identification and detection issues. The improved use of the network's processing capacity is the key differentiator of this design. It is essentially a 27-layer convolutional neural network (CNN). The model overview is given below in figure 32 (Szegedy et al.,2014)

Figure 32

Architecture of Inception Net





Note. Architecture of Inception Net adopted from "*Structure of Inception Net*" by Ye Ni. https://ieeexplore.ieee.org/document/9139344

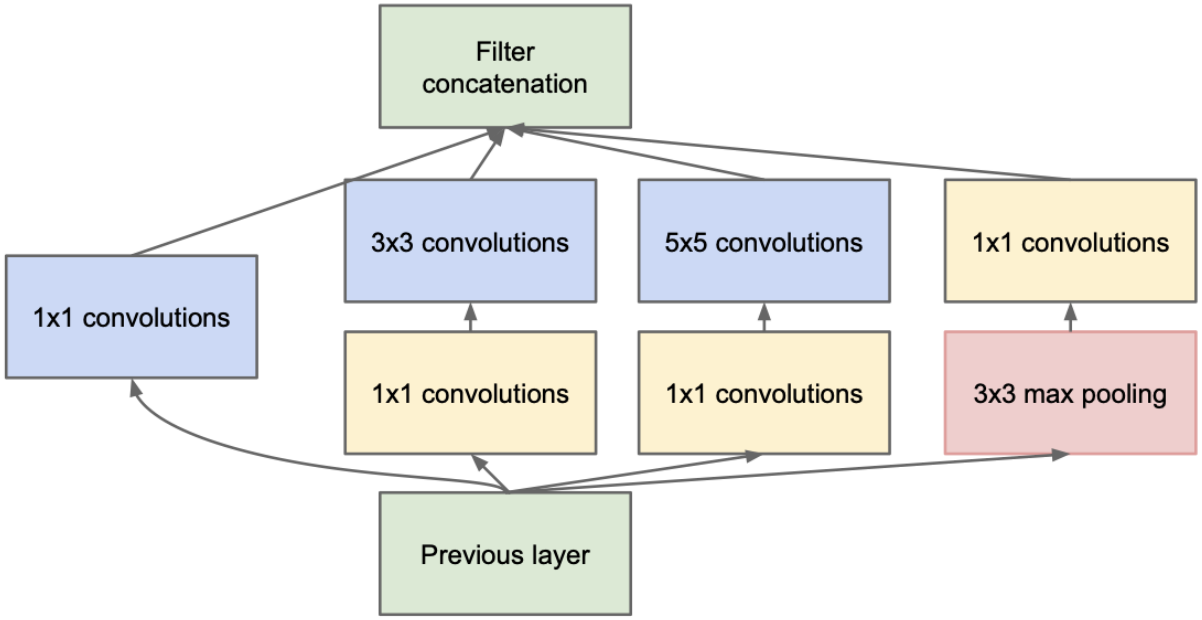
In the above figure 32 there are few layers which are called Inception layers. The following stage input is formed by concatenating the output filter banks of all those layers namely, the 1x1 convolutional layer, the 3x3 convolutional layer, and the 5x5 convolutional layer into a single output vector. In addition to the layers indicated above, the original inception layer also has two significant additions. Prior to adding another layer, a parallel max pooling layer and an 11-convolutional layer that are often employed for dimensionality reduction provide the inception layer with more options.

As seen in figure 33 below, an Inception network is often constructed from modules of the previously mentioned type stacked on top of one another, with intermittent stride 2 max-pooling layers added to reduce. It looked advantageous to start employing Inception modules exclusively at higher layers while leaving the lower layers in the conventional convolutional form for technical reasons (memory efficiency during training). This only reflects certain infrastructure inefficiencies in our present approach and is not absolutely essential. This architecture's ability to rapidly increase the number of units at each step without causing an uncontrollable explosion in computing complexity is one of its key advantages. Because

dimension reduction is so widely used, it is possible to protect the many input filters from the final stage from the following layer by first lowering their dimension and then convolving over them with a big patch size.

Figure 33

Inception module with dimension reductions



Note. Inception Net module adopted from "*Inception module with dimension Reductions*" by Christian Szegedy. https://arxiv.org/abs/1409.4842.

Enhanced ResNet

We will also implement a model which will be modified from the suggested base model of ResNet. We picked ResNet since it is the newest model out of the four. We thus implement two changes to further improve it. Integrating the SE module is one of them. The Squeeze & Excitation (SE) module, which explicitly models channel interdependencies, autonomously reorients channel-wise feature responses to optimize model performance and the addition of focus loss is the other change. As a consequence, the linkages and prominence of the channels

may be efficiently leveraged. The other is employing focal loss, which may effectively address the class imbalance problem of having a larger proportion of low-level intensity pixels than high-level intensity pixels. As a result, the ResNet model might perform better by extracting characteristics and predicting poverty after adopting two enhancements.

Random Forest

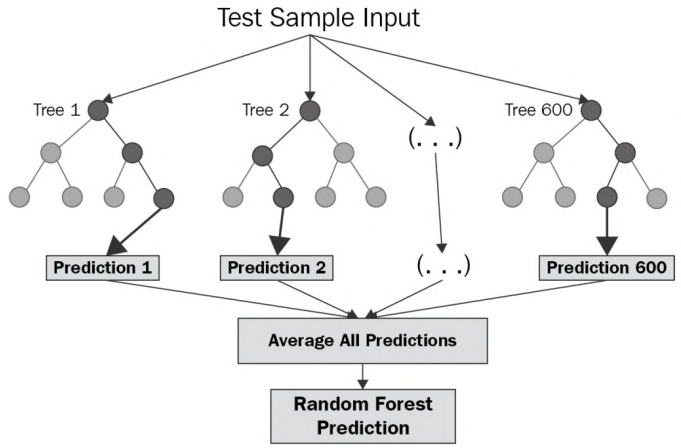
Classification and Regression Trees (CART) are a subset of decision trees that comprise both classification and regression trees. Random Forest demands more processing during training because of the high number of decision trees it includes. We discussed how to predict the target characteristic using the machine learning approach Random Forest. In order to determine which model best evaluated the target feature, we will also contrast this one with others like Linear Regression, Lasso, and Ridge. The outputs of the model are improved by creating more models because Random Forest is an ensemble method. It is overemphasized that training numerous trees on a single dataset in the Random Forest creates comparable trees. This might result in a substantial tree connection. To address this problem, the training dataset is subjected to bootstrap sampling. For the Random Forest approach, samples from the initial dataset were chosen at random. The technique in issue is referred to as bootstrapping. The size of this bootstrap dataset should match that of the original dataset. Because replacement is used in random sampling, the sample instance may be used several times. No two decision trees will have the same training set. Bagging is the first of the three different ensemble methods. Bagging and boosting are two further strategies that lessen prediction variance by using additional data for training from the original dataset. By lowering the high bias that affects the models, boosting helps to increase their complexity. It is possible to produce more random trees by using a random threshold for

each feature, similar to how a Random Forest chooses a random subset of features. This band of wildly random trees is known as Extremely Randomized Trees.

As a consequence, the extra trees require less processing time during training compared to a typical Random Forest (Géron, 2022). Random samples with random attributes are provided to each decision tree for training from the test sample input dataset. As seen in figure 34, there are 600 decision trees, and a random sample is given to each one of them. These samples are also known as bootstrap samples. The 600 outputs produced by the decision trees are then averaged and totalled. The output from the Random Forest that is still present after averaging its true output.

Figure 34

Random Forest Overview



Note. Random Forest diagram was taken from Chakure's 2022 article "*Random Forest Regression - The Startup.*"
https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f

Linear Regression

When endeavoring to determine the relationship between two continuous variables, simple Linear Regression is helpful. An independent variable makes a prediction, whereas a

dependent variable makes a reaction. It seeks statistical correlation rather than deterministic correlation. If a relationship between two variables can faithfully reflect only one of them, it is said to be deterministic. Statistical correlation cannot provide an accurate assessment of the relationship between two variables.

Finding the line that most closely matches the data is the main goal. The lowest potential total prediction error across all data points is considered to be the best fit line. The term error refers to the separation between a point and a regression line.

One may find a linear relationship between the objective and one or more variables using Linear Regression. Simple and multivariate Linear Regression are the two varieties. In mathematics, simple Linear Regression is denoted by the expression (2).

$$Y \approx \beta_0 + \beta_1 X Z \tag{5}$$

$$\text{RSS} = (y_1 - \widehat{\beta_o} - \widehat{\beta_1} x_1)^2 + (y_2 - \widehat{\beta_o} - \widehat{\beta_1} x_2)^2 + \ldots + (y_n - \widehat{\beta_o} - \widehat{\beta_1} x_n)^2 \tag{6}$$

The concept of a finest line, or the line that most closely resembles the provided data, is frequently used in regression. The loss function or cost function is what would lead to a good fit or accuracy in this context. The least squares approach is the name of this strategy. Using some arithmetic, the least squares approach selects 0 and 1 to lower the RSS. A new set of coefficients is then produced, and we must follow particular procedures to ensure that these estimated coefficients are accurate.

Lasso Regression

Shrinkage is used in the Linear Regression approach known as Lasso Regression. When data values move closer to a middle value, such as the mean, this is referred to as shrinkage. The Lasso approach favors models that are basic and sparse (i.e. models with fewer parameters). This

sort of regression is ideally suited when models exhibit significant levels of multicollinearity or when critical components of the model selection approach, such as variable selection and parameter removal, need to be automated.

L1 regularization is used in Lasso Regression, which results in a penalty proportional to the absolute size of the coefficients as shown in (7). This form of regularization may produce a sparse model with few coefficients, certain coefficients may become zero and be removed from the model. Greater penalties result in coefficient values that are closer to zero, which is ideal for creating simpler models. However, L2 regularization does not eliminate coefficients or sparse models (like Ridge Regression). As a result, the Lasso is simpler to grasp than the Ridge. Lasso solutions are computer-assisted techniques to quadratic programming issues (like Matlab).

$$\sum_{i=1}^{n}(y_i - \sum_j x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p}|\beta_j| \tag{7}$$

This is equivalent to minimizing the sum of squares with constraint β_j in summation notation. Some of the x_{ij} are lowered to absolutely zero to provide a regression model that is easier to comprehend.

Ridge Regression

Ridge Regression is a form of Linear Regression approach used in machine learning. When there are several highly correlated variables, Ridge Regression is utilized. Ridge Regression requires the addition of a penalty term to the cost function that is proportional to the sum of the squares of the coefficients. It helps to avoid overfitting by penalizing the variable coefficients. Ridge Regression reduces overfitting by including a penalty term in the error function that restricts the magnitude of the coefficients. As a result, the coefficients get lower and the optimization issue gets simpler to solve. This penalty term pushes the model to strike a

balance between being simple and having a good fit to the training set of data. This may be useful when there is a lot of noise in the data since it prevents the model from being extremely sensitive to certain data points. Ridge Regression can therefore help to improve the generalizability of a machine learning model. Ridge Regression is typically used with other machine learning techniques such as cross-validation to further reduce overfitting. Ridge Regression has the drawback of being computationally demanding and requiring more data to produce appropriate findings. Equation 8 and equation 9 shows the regularization term used in ridge regression and also the cost function.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)} + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2 \quad (8)$$

$$RSS_{ridge}(w,b) = \sum_{i=1}^{n} (y_i - (w_i x_i + b))^2 + \alpha \sum_{j=1}^{p} w_j^2 \quad (9)$$

This is how the cost function for Ridge Regression appears as shown in the above formula. The cost function has two functions, as you may have noticed. The first is the same cost function as the Linear Regression model. This term guarantees good data fit for training. The L2 penalty or regularization term is the second term. The aim of the term is to keep the parameters low. When x is the input value, y is the projected value, λ is the penalty term, and β is the coefficient. You can see that the error term is increased by the penalty term. We want to reduce the inaccuracy and the magnitude of the coefficients in Ridge Regression. We are prompted to strike a compromise between these two goals by adding the penalty term.

Model Supports

Finding a platform where we can construct these models will be essential for us to achieve the best outcomes for predicting wealth in Rwanda and for promoting awareness of

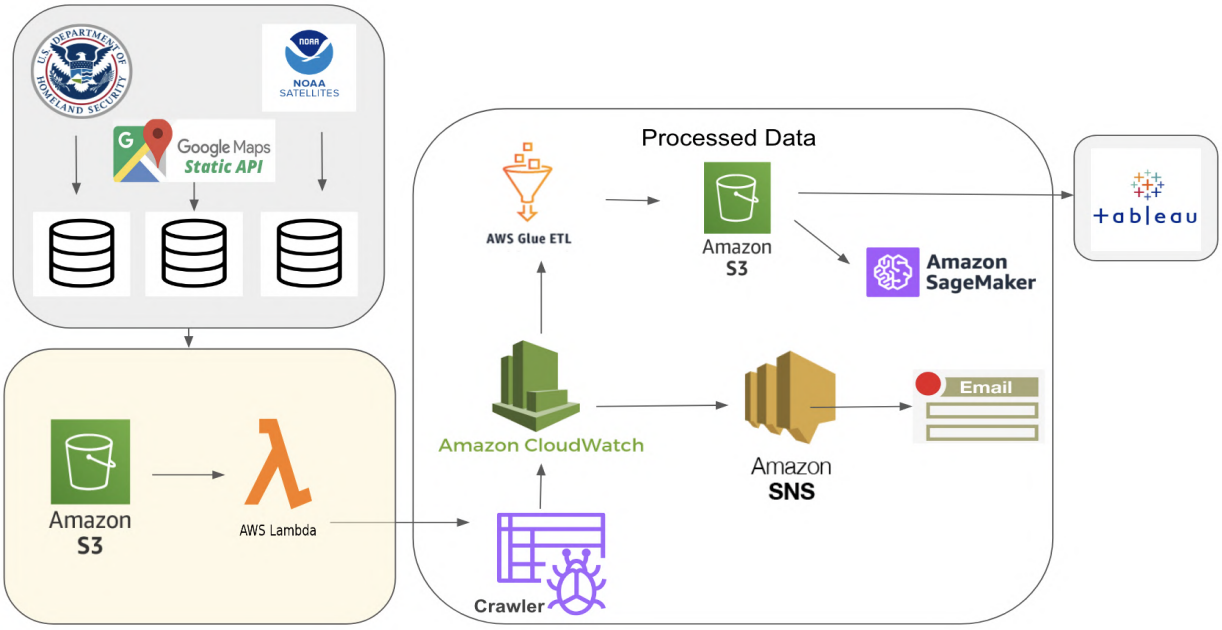
developing regions like these. The size of the dataset is a key factor in choosing the platform that will assist the development of our models. The dataset can be kept locally on the local system as one approach the information can then be added to the jupyter notebook in order to construct the models. We will use the tool jupyter notebook to provide an interactive environment with an intuitive user interface, however there is a drawback to this approach: storing data on a local system can be a major problem. Data loss, laptop loss and exposure to others are a few of these problems. Our team will use amazon web services to solve this issue instead many businesses use the cloud platform Amazon Web Services for their infrastructure and applications. Now that we have decided to use aws as our platform our next step is to pick which environment we will use to execute our models. The AWS SageMaker environment will be our ideal environment as we explained in our Data Management Plan. Our team will work together in the SageMaker environment to do machine learning and deep learning on our models using Jupyter notebook, which runs locally inside of SageMaker.

The steps of running a scalable model on the AWS platform need forethought. The first step is to build an S3 bucket, which is a cloud data lake for storing data. We would next upload text and picture data from the local system to S3. For services like S3 to communicate with one another, we'd need to create an IAM role, which also stands for IAM identity. This enables services to communicate with one another. AWS Glue is an extract, transform, and load (ETL) tool that we will use to crawl our data from S3 and transform it before using it in the Jupyter notebook. To automate this process we will write a function in lambda which will detect data from S3 and load into Glue to perform ETL. To make a full scale pipeline of our AWS platform we will also use AWS CloudWatch, which is used to monitor cloud resources and we will connect it to AWS SNS which will send us an email if there is a bug within any of our resources

while flowing through the pipeline. Finally, we will also connect Tableau to S3 to perform visualizations of processed data as shown in the figure 35.

Figure 35

Illustrates our pipeline for amazon web service (AWS)



SageMaker has a number of instances, ranging from CPU to GPU, which influences speed. Due to the large amount of data from many sources, using the CPU will be time consuming. GPUs are utilized for quicker computing during the training and testing of the face mask detector model. SageMakers Accelerated computing instance will be used to work on the model. This contributes to more comfortable features and capabilities and a better interface for modeling purposes. SageMaker ml.p3.8xlarge with 4 GPU 32 vCPUs and 244 GiB is used specifically for training and testing. We will run 150 hours overall for the project, which will consist of training, validation, testing, and so on. Figure 36 shows the architecture for running models in SageMaker.

Figure 36

Illustrates the AWS architecture for running models in SageMaker.

Jupyter notebook supports many languages such as Java, R, Julia, Matlab, Python, etc. Python is one such programming language that has several tools for doing different preprocessing and machine learning processes. Scikit-Learn is a Python machine learning package that contains numerous machine learning algorithms as well as various assessment metrics and ways for modifying hyperparameters. Matplotlib and Seaborn libraries are used to visualize data and obtain insights by producing various sorts of graphs. Jupyter notebook is popular among them because it is a cell-based IDE that allows each cell to be performed independently. It's also more feasible for us as Jupyter notebook is integrated within SageMaker and allows us a lot of working freedom. The tools and environments used for this project are mentioned in table 5

To understand more about the data, we will do exploratory data analysis within the Jupyter notebook using the Seaborn and Matplotlib programs. Using Pandas and NumPy techniques, the data may be cleaned by eliminating any outliers, missing values, and duplicate

values. The train test split function divides the data into train, validation, and test segments with percentages of 70%, 10%, and 20% respectively. As a result, the model is trained on training data and can predict on testing data. The evaluation is carried out to assess the model's performance. Figure 37 shows the flow of data for the project throughout the data collection phase to data modeling phase.

Figure 37

Data Flow From Data Collection to Modeling

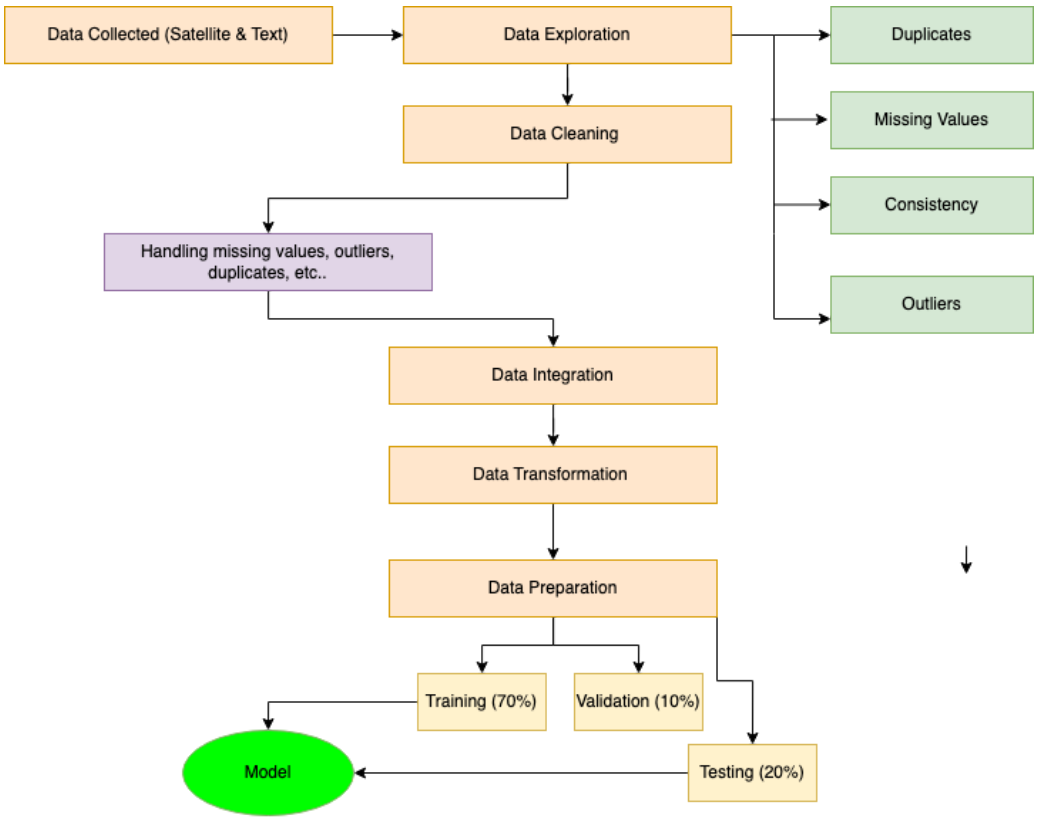


Table 5

Tools and Environment required

Resource Function	Resource Type	Resource Detail
Jupyter Notebook	Platform	Interactive platform to run our code.
SageMaker	Environment	Environment to perform machine learning and deep learning on our models.
AWS S3	Tool	Data Lake to store all our data.
AWS Lambda	Tool	Lambda function to automate data crawling for our data.
AWS Glue	Tool	Glue platform to help us perform ETL on our data.
AWS CloudWatch	Tool	CloudWatch to help us monitor our cloud resources.
AWS SNS	Tool	To send a message to our email if there is a bug within our pipeline.

Note. Technologies and tools required along with its description

Model Comparison and Justification

Our project goal is to predict the wealth of a location using the satellite images, nightlight images and survey dataset for the location. We are going to use deep learning CNN models on the images and also use a few regression models to see the performance of the survey dataset in predicting the wealth of a location using the information generated by the images. Model comparison is done in different categories, depending on the problem. Regression models for predicting wealth from survey data and models using satellite images. The machine learning models for regression include Random Forest, Lasso, Ridge, and Linear Regression.

Linear Regression

Linear regression is a simple regression technique where using the dependent variables, we predict the independent variable. The goal is simple, to consider the given dependent variable and plot the best fit line to fit the model. The advantages of linear regression are that it works well when the data is linearly separable. It is simple to apply and analyze and it can handle model overfitting using a variety of strategies such as dimensionality reduction regularization techniques and cross validation. The cons of linear regression is often prone to overfitting and noise in data, it is very sensitive to outliers and also prone to multicollinearity in data.

Ridge Regression

Ridge Regression is a specific method for analyzing multicollinear multiple regression data. Although it is a key regularization approach, it's complicated science prevents it from being extensively employed. Using the shrinkage parameter λ (lambda), Ridge Regression is a method for avoiding multicollinearity-related data complexity. The observed value differs much from the real value despite the fact that the least squares estimates (OLS) in multicollinearity are unbiased. By slightly biasing the regression estimates, Ridge Regression reduces the standard errors. The advantages of Ridge Regression is that it helps in tackling overfitting of a model, trades variance for bias in presence of collinearity in data. The disadvantage of Ridge Regression is the model interpretation is low and we need to select hyperparameters.

Lasso Regression

Lasso Regression is a technique in which shrinkage is used in Linear Regression. Shrinkage happens when the data values are shrinking towards a middle value. Models which have lesser parameters are more preferred for Lasso approach. When massive multicollinearity is observed or when you want to automatically select models that feature variable selection and

parameter elimination for model selection then Lasso Regression is more preferable. Most of the time this model produces models with less coefficients and there is a chance that certain coefficients may run to 0 and thus be removed from the model. Lasso Regression can generate coefficients with values closer to 0 which helps create simpler models.When we compare Lasso Regression to Ridge Regression, we can notice that we don't observe removal of coefficients. This makes Lasso Regression easier to understand and implement compared to Ridge Regression. It provides more performance by being fast and efficient compared to Ridge Regression. Connected variables and other variables are put to zero as Lasso Regression only keeps correlated variables. When we have correlated variables, the fundamental issue with Lasso Regression is that it only keeps one variable and puts the other connected variables to zero. This causes information loss and causes reduced model accuracy.

Random Forest

We will be looking at a Random Forest algorithm which is a supervised learning algorithm which incorporates multiple decision trees and creates a forest. The functioning of a Random Forest is rather simple and even after that it is able to achieve higher accuracy without any hyperparameter modifications. Random Forest takes multiple outputs from various trees into account and combines the outputs to an accurate result. Feature selection is done implicitly in a Random Forest algorithm. It uses random feature selection to generate this result. Ultimately results in producing data with articulating features. Outliers don't have much effect on the results when working with Random Forest algorithms. We can bin the variables to get to this result. It can handle linear as well as non linear relations. It is preferred for clean accuracy and also provides a balanced bias-variance trade off. The model’s purpose is to take an average output value for all the outcomes that are being generated and thus it is good for averaging the

outcomes. The only issue is how difficult these models are to understand. It doesn't provide full readability into the model. It also doesn’t work with large datasets as it costs a lot of computing power. We don t get a lot of control over the model behavior when we are using random forest algorithms. We used VGGNet, DenseNet, ResNet and InceptionNet to obtain information from the images. We used CNN to obtain luminosity information from images.

VGGNet

VGG net uses CNN which helps with accurate image identification as it uses small CNN features to perform the task. It also won the image net challenge internationally. An advanced VGG net can compute a maximum of 19 layers within itself. VGG is a deep CNN and still performs better compared to ImageNet when images outside the dataset are compared. Number of layers used increases to twice as we go deeper. This encouraged the production for VGG16. One of the only downpoints is how much time it takes to train parameters. The complexity of VGG16 makes it a larger file and it takes up to 533 mb size. This increases the computing time of the VGG network.

InceptionNet

When earlier models were only going deeper to improve performance and accuracy but compromising the computational cost, Inception Net accomplished a milestone in CNN classifiers. On the other hand, the Inception network has advanced engineering. It employs numerous techniques to boost performance in terms of speed and precision. In an inception network, modules are repeated, known as inception modules to make the network deeper. Fewer parameters, improved computational performance, and multiscale feature extraction make Inception-Net superior to VGGNet. The inception v1 model should aid in lessening the impact of the vanishing gradient problem, but the authors of the article discovered that during the early

stages of training, this classifier didn't significantly enhance convergence. You will waste a lot of processing resources on these because of their poor initialization (C. Szegedy *et al.,* 2015). Because the input dimensions are reduced, which greatly increases the risk of information loss, the use of 5x5 filters in Inception v1 results in a reduction in accuracy. Inception V2 provided a solution to this issue.

ResNet

Residual Network, which supports Residual Learning, is known by the abbreviation Resnet. The 50 denotes how many layers there are in it. Resnet50, thus, is an acronym for a residual network with 50 layers. For image classification, deep CNN has produced a lot of innovations. The basic trend is to add additional layers as you solve complex problems and improve classification and identification precision. However, when we use neural networks to learn more, accuracy begins to saturate and eventually declines. Residual training makes an effort to address this issue. An extensive number of layers are stacked and trained for the given job in a deep CNN. Instead of attempting to learn certain features, residual learning aims to learn some residual features. Residual can be easily regarded as the feature learned from that layer's input subtracted. ResNet does this utilizing shortcut connections, or the direct connection of input from one (n+x)th layer to another. It has been demonstrated that training these networks is simpler than training straightforward deep convolutional neural networks, and the issue of accuracy degradation is also addressed.

DenseNet

DenseNet contrary to ResNet's proposal to deepen the network and Inception-proposal to extend the network, DenseNet suggests reusing the multilevel features by dense shortcut connections between layers. ResNets have a large number of parameters since each layer needs

to learn its own weights. Instead, DenseNets layers only add a tiny number of new feature-maps and are extremely constrained. DenseNets offer a variety of appealing benefits, including the elimination of the vanishing-gradient issue, improved feature propagation, promoted feature reuse, and significantly fewer parameters. It does, however, have some drawbacks. A L-layer DenseNet has a total of $L(L-1)/2$ connections. In addition to lowering the computing and parameter efficiency of networks, too many connections can increase their propensity for overfitting.

In table 6 , we noted down the strength and weaknesses of each model which we are going to implement in our project.

Table 6

Model comparison

Model	**Strength**	**Weakness**
VGG Net	• Structure simplicity • Good performance in image classification	• slow to train • network architecture weights themselves quite large
Inception Net	• Stack 3 different size of kernels • Better than VGG as it widens the network with less error	• Repetitions of components • lot of resources required because of poor initialization
ResNet	• Train without difficulty on	• Need more time and extra

Model	**Strength**	**Weakness**
	deep network ● Prevents model from overfitting/underfitting	resources to train
DenseNet	● Different from InceptionNet & ResNet as multilevel features for short dense connections	● Compromise parameters efficiency of networks ● May lead to overfitting
Enhanced ResNet	● Improves base ResNet Model by integrating SE module & Focal Loss ● Good performance in image classification	● Requires extensive computational resources
Linear Regression	● Works well when data linearly separate	● Prone to overfitting & noisy data ● Prone to multicollinearity of data
Lasso Regression	● Less coefficients as compare to Ridge Regression ● Performs faster and more	● May lead to information loss ● Reduces model accuracy due to less variables

Model	Strength	Weakness
	efficient than Ridge	
Ridge Regression	• Avoid overfitting of model • Reduces the number of standard errors	• Interpretation of model is low • Need selected hyperparameters
Random Forest	• Achieve good accuracy without hyperparameter tuning	• Does not work properly with large datasets

Note. Table showing strength and weakness of each models required

Model Evaluation Method

The main step once the model is built is to assess it to determine how well it works. Even when a model is constructed, it must be tested against test data to determine various metrics that provide information about the various performances of the model and offer us a suggestion on which model to apply to our data. When evaluating the performance of the final models using the various metrics, the best model is chosen during the model evaluation step. In order to evaluate both the deep learning (VGG 16, ResNet, InceptionNet) and machine learning (Linear Regression, Lasso, Ridge, Random Forest) models we will be using some evaluation metrics such as F1 score, Precision, Recall, R2, Mean Absolute Error (MAE) (Manas et al. 2018) and Root Mean Squared Error (RMSE) (Madhur et al. 2021). We need to evaluate all these in order to get the best model for poverty prediction. Precision (10), recall (11), and F1-score (12) are the measures we compute to assess performance.

$$\text{Precision} = \frac{TP}{TP+FP}$$

(10)

$$\text{Recall(Sensitivity)} = \frac{TP}{TP+FN}$$

(11)

$$\text{F1-Score} = \frac{2*precision*Recall}{Precision + recall}$$

(12)

TP stands for true-positive predictions, FP for false-positive prediction, and FN for false-negative prediction. Recall quantifies the portion of positive samples from the ground truth that are labeled as shown in (9). According to precision as shown in (8), the proportion of samples that were expected to be positive are in fact positive. An algorithm's accuracy is determined by the proportion of correctly classified predictions (TP) to all classifications that are expected to be positive. Despite the model's high accuracy, the precision score is crucial in determining if it accurately predicts a given goal value. The two criteria are combined to get the F1-score as shown in (10), which represents overall performance. The performance is better the higher the three measures are.

R^2 *(Coefficient of Determination)*

Moreover, in order to assess how well poverty can be predicted, we use the coefficient of determination r2 as shown in (13), which is defined as

$$r^2 = \frac{\Sigma(\hat{y}_i - \bar{y})^2}{\Sigma(y_i - \bar{y})^2}$$

(13)

If y_i is the actual poverty index for the *i*th test sample, $\widehat{y_i}$ represents the projected poverty index, $\overline{y}$ and denotes the mean of all the actual poverty indices. A greater value of the determination coefficient denotes better performance and has a range of [0, 1].

RMSE

The standard deviation of prediction errors, often known as residuals, is measured using the RMSE formula. It is a measurement that depicts the distribution of these residuals in the prediction. The equation (14) provides a solution.

$$RMSE=\sqrt{\frac{1}{n}\Sigma(S_i - O_i)^2} \tag{14}$$

Where S_i stands for the predicted values of a variable, O_i stands for the observations, and n is the maximum number of observations that can be employed for analysis. Since RMSE depends on scale, it should only be used to compare the accuracy of different models or model configurations in predicting a single variable, not between variables. Mean absolute error (MAE) is a popular statistic since the error value units correspond to the anticipated target value units.

MAE

Unlike RMSE, MAE changes are logical and linear. The squaring of the error value by MSE and RMSE, which punish larger errors more severely, causes the mean error value to be inflated or raised. In MAE, the scores increase linearly as the number of errors increases rather than being given a different weight for each type of error. The MAE score is determined by averaging the absolute error statistics. A negative integer becomes positive through a mathematical process known as the absolute. The formula described below (15) is used to calculate the MAE.

$$\text{MAE}= \frac{1}{n}\Sigma\left|y_1 - \hat{y}_i\right|^2 \tag{15}$$

Model Validation and Evaluation Results

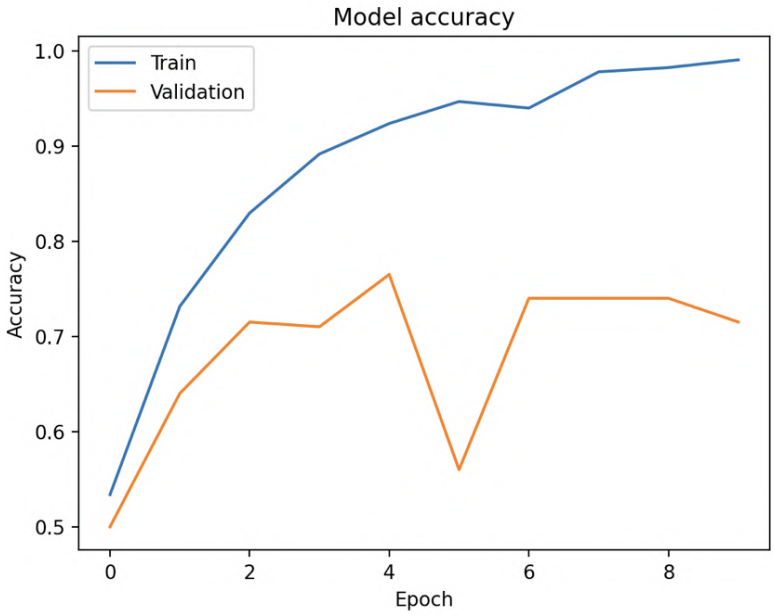
The model validation and evaluation was done by applying the trained models on the validation data and then predicted on the test data. The validation of the models and the evaluation of the models is done by using the Sklearn library in python.

In this project, for the deep learning models, the validation for the models was done by using validation split data which was 10% of the total training data and k-fold validation was implemented.

By using the k-fold validation, instead of splitting the data in a single split, multiple folds are used. The training accuracy for the VGG - 16 model, 99% accuracy was achieved and the validation and testing accuracy achieved was 71%. The figure 38 shows the training and validation accuracy. The model is trained for 10 epochs.

Figure 38

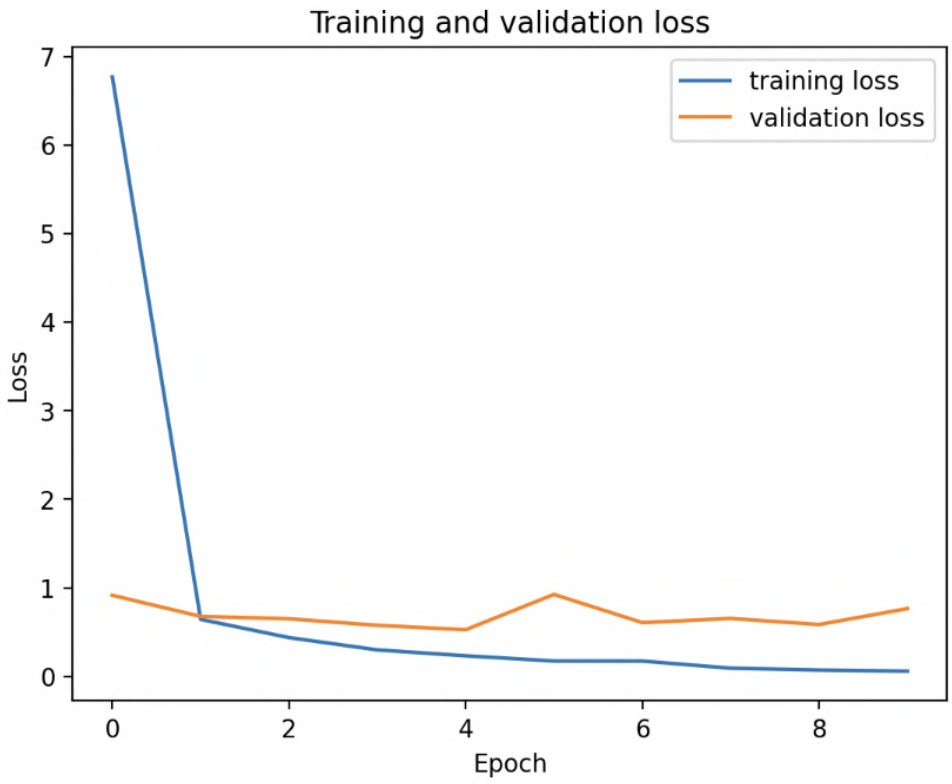
Training and Validation Accuracy for VGG-16 Model



The training and validation loss for the VGG-16 model was 0.0584 and 0.7150. The figure 39 shows train and validation loss for the model. We observed a sudden drop in the training loss after the first epoch and then it decreased gradually through the epochs.

Figure 39

Training and Validation Loss for VGG-16 Model



Another evaluation metric used was confusion matrix, we plotted a confusion matrix for the three classes in our data. The model's predictions are compared against the true labels across three classes.From the confusion matrix , we can derive various performance metrics such as accuracy, precision, recall, and F1-score for each class based on the counts in the matrix. These metrics provide insights into the model's performance for individual classes and overall. For the VGG-16 model, we can see that the model accurately identified 96% of the data from the dim class, 46% from the medium class and 62% from the bright class. Figure 40 shows the confusion

matrix for the VGG-16 model, where the true labels are plotted on the y - axis and the predicted labels are shown on the x - axis. The values on the diagonal are the true predictions made by the model.

Figure 40

Confusion Matrix for VGG-16 Model

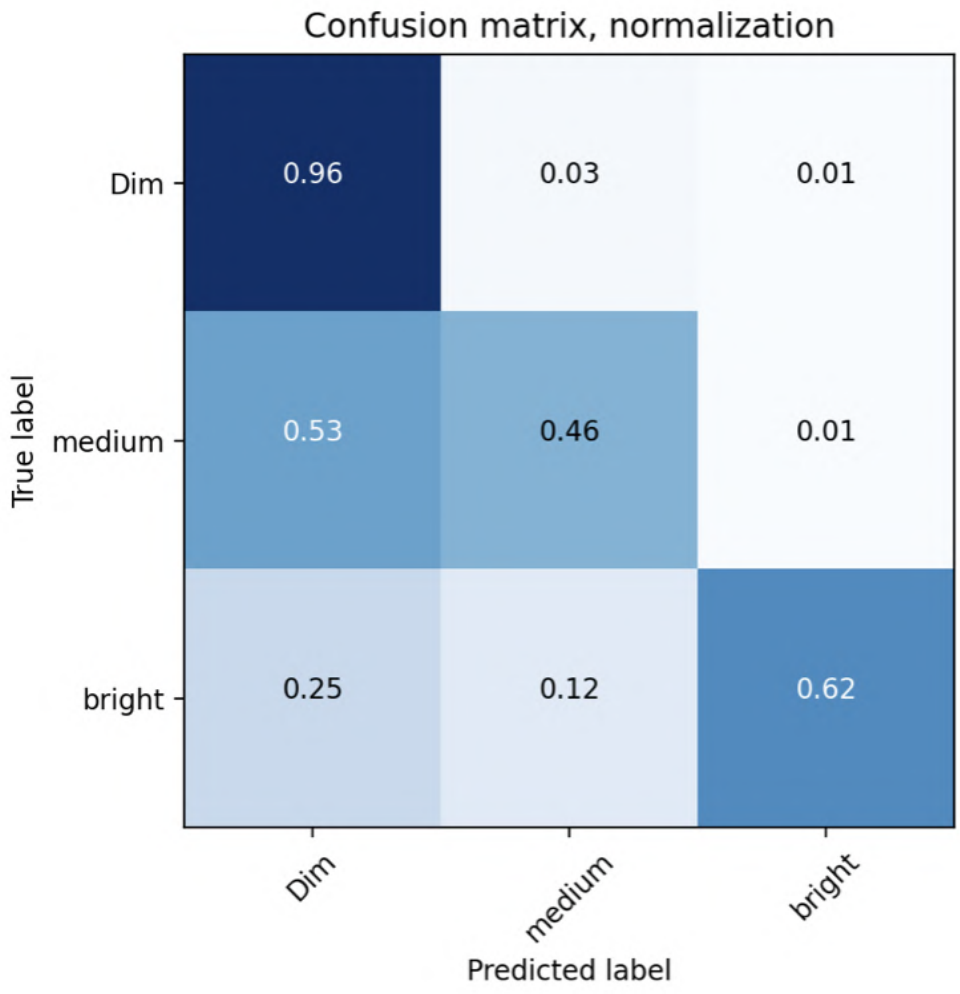


Figure 41 shows the classification report for the model. A classification report is a useful summary of the performance of a classification model, providing metrics such as precision, recall, F1-score, and support for each class. It helps to evaluate the model's accuracy and effectiveness in classifying different classes in a multi-class classification problem. The accuracy

value represents the overall accuracy of the model in correctly classifying all classes. The accuracy for our trained model was 71%.The macro average and weighted average provide a summary of the precision, recall, and F1-score across all classes, taking into account the class imbalance if any.

Figure 41

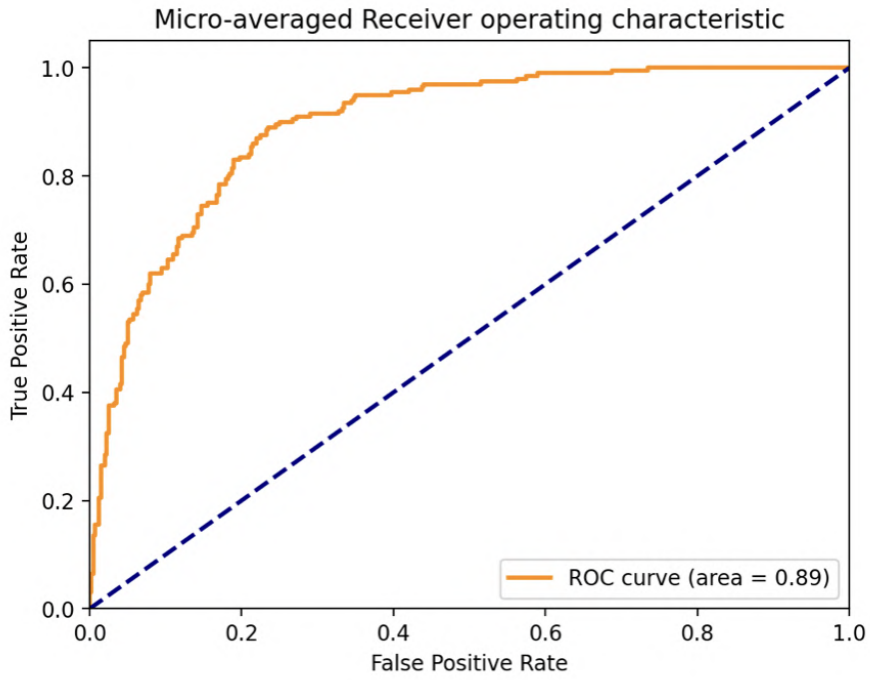
Classification Report Of VGG-16 Model

	precision	recall	f1-score	support
dim	0.65	0.96	0.78	100
medium	0.91	0.46	0.61	92
bright	0.71	0.62	0.67	8
accuracy			0.71	200
macro avg	0.76	0.68	0.68	200
weighted avg	0.78	0.71	0.70	200

Based on the figure 42 as shown below which represents ROC curve and AUC score, we can see that the model has a good receiver operator characteristics(ROC) curve as it is more towards positive y axis and good area under curve(AUC) value of 0.89.

Figure 42

ROC Curve and AUC Score of VGG-16 Model



For the innovative model, where we integrated a SE module into the VGG-16 model, The model showed an improvement from the baseline model of VGG-16. The improved accuracy for the model on training data was 97.29% and the validation accuracy improved to 77%. Figure 43 shows the training and validation accuracy for the innovative model. During the 10 epochs of training the model accuracy improved constantly and similarly the validation accuracy also improved for the model. The figure 44 showed the loss graph for the training and validation phase for the model. The training loss at the end of the training was reported to be 0.0924 and the validation loss was 0.5645.

Figure 43

Training and Validation Accuracy for VGG-16 Model with SE Module

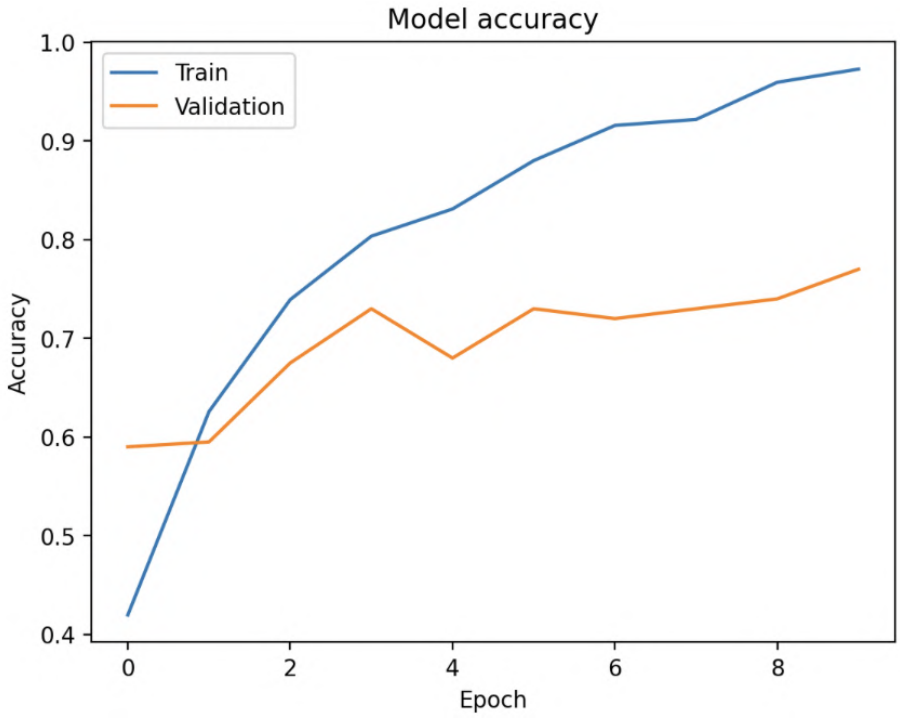


Figure 44

Training and Validation Loss for VGG-16 Model with SE Module

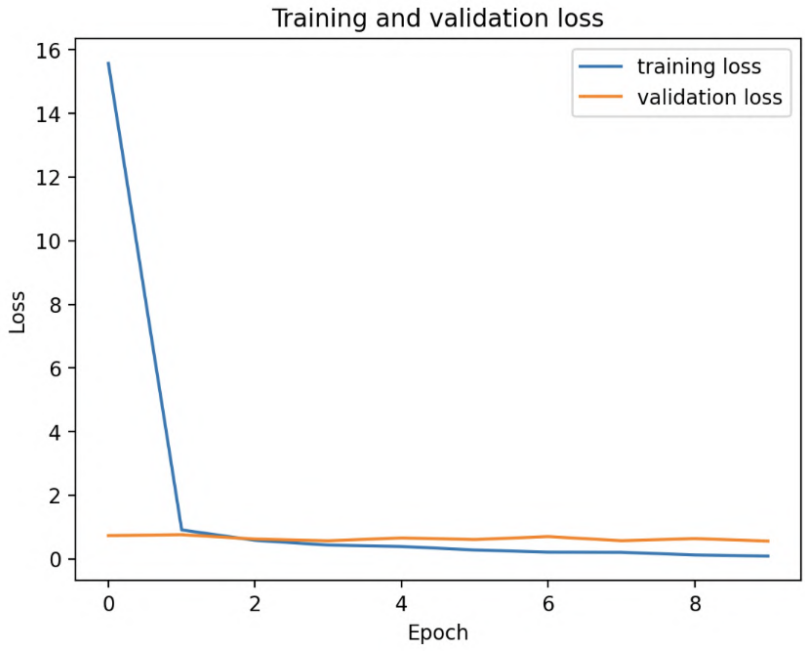
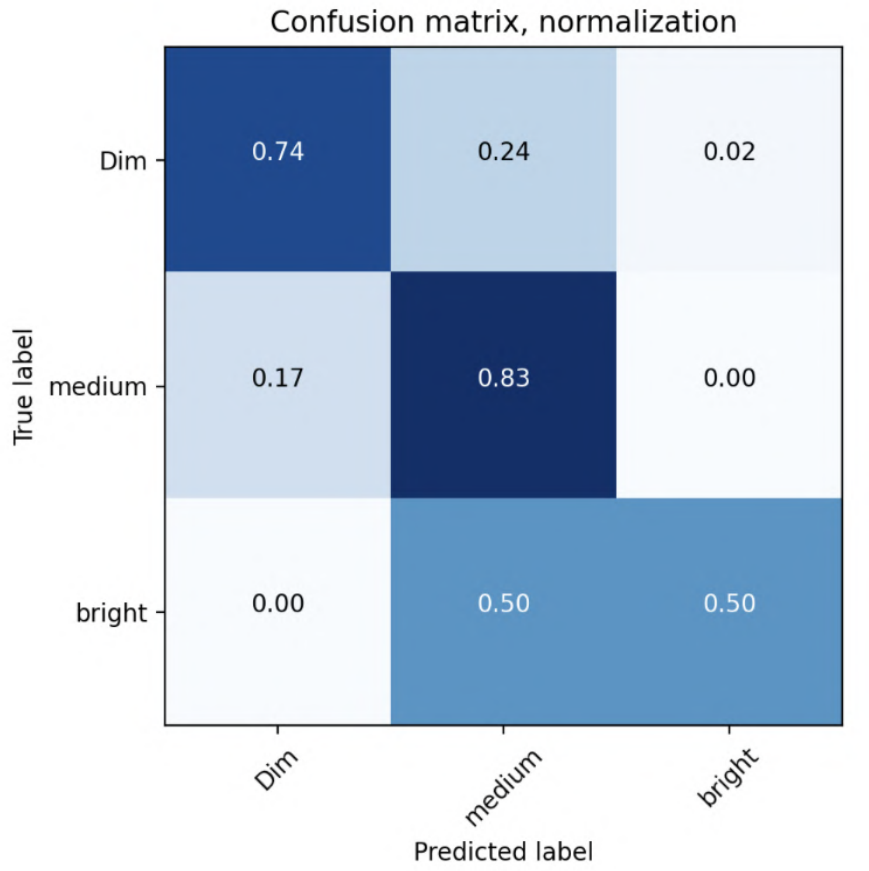


Figure 45 shows the confusion matrix for the VGG-16 model with SE module. Compared to every other model which we trained for our experiment, the confusion matrix was better as it gave balanced accuracies for each class instead of favoring anyone class.

Figure 45

Confusion Matrix for VGG-16 Model with SE module



Also the accuracy for the model was 77% which was the best accuracy out of all the models with a recall value of 0.77 as shown in the figure 46. Also has shown in the figure 47 the AUC has a value of 0.91 and also the ROC is towards the positive y axis which is a good sign

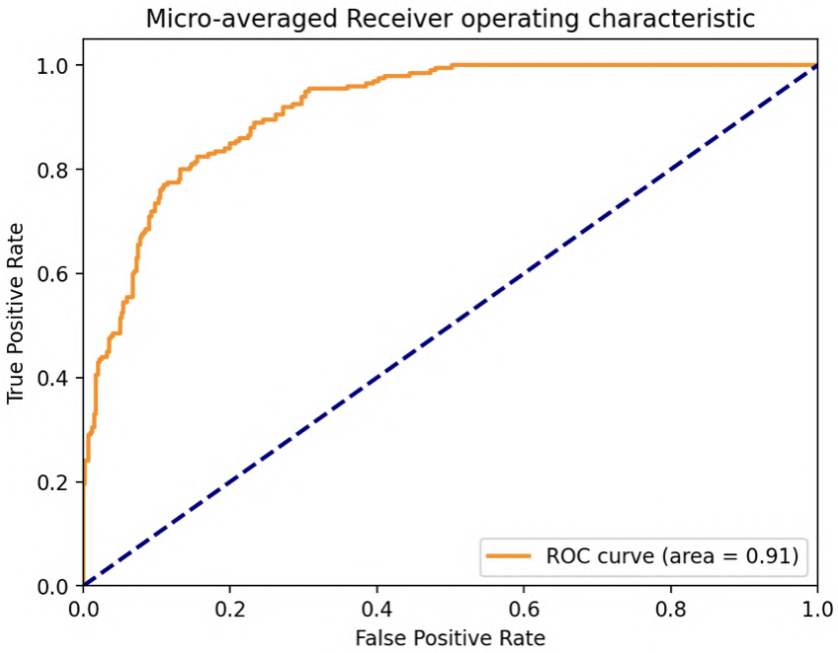
Figure 46

Classification Report Of VGG-16 Model with SE Module

	precision	recall	f1-score	support
dim	0.82	0.74	0.78	100
medium	0.73	0.83	0.78	92
bright	0.67	0.50	0.57	8
accuracy			0.77	200
macro avg	0.74	0.69	0.71	200
weighted avg	0.77	0.77	0.77	200

Figure 47

ROC Curve and AUC Score of VGG-16 Model with SE Module



For the ResNet Model, the overall accuracy for the model was 76% which was similar to the previously discussed model. From the figure 48 which shows the training and validation accuracy curve. The training accuracy for the model was 93.48% and the validation accuracy was 77% while training loss value is 0.1794 and the validation loss value is 0.4884. The loss curve for the training and validation is shown in the figure 49.

Figure 48

Training and validation accuracy for ResNet Model

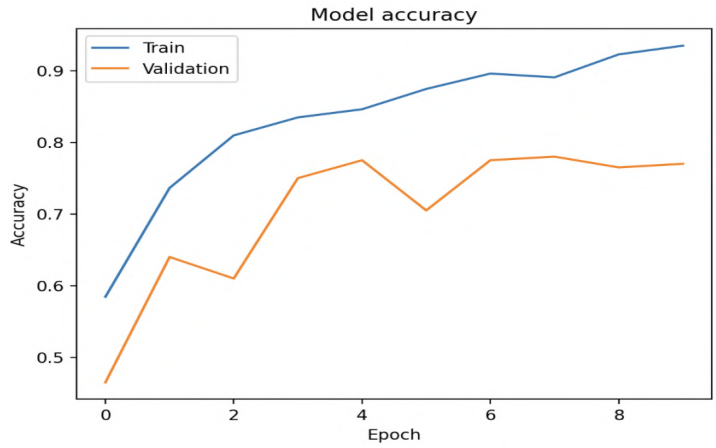


Figure 49

Training and validation loss for ResNet Model

Confusion matrix for the ResNet model is given in figure 50 where it shows very good performance in the medium and bright class but shows not so good performance in the dim class. For the dim class, the model was able to correctly classify 67% of the data. For the medium class, the model performed very well as it predicted 88% correct images and for the bright class it predicted 75% correct images. The precision for the bright class was of value 1. It is shown in

figure 51 which shows the classification report for the ResNet model. The recall value for the model is 0.77

Figure 50

Confusion Matrix for ResNet Model

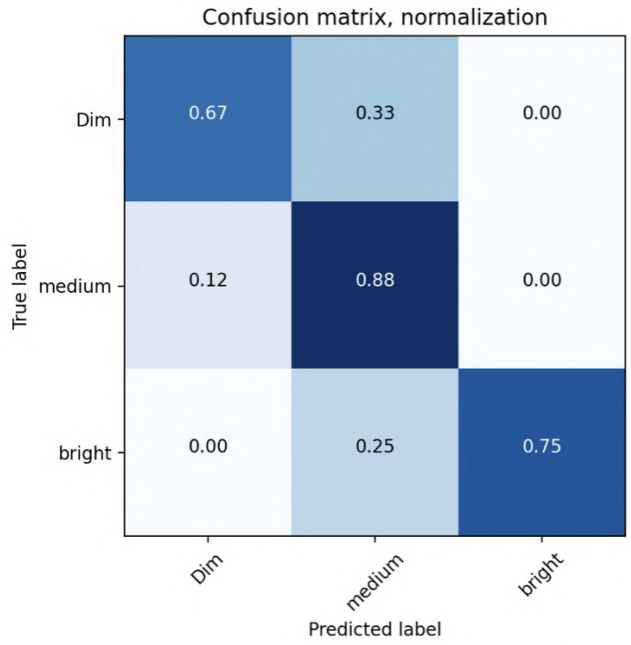


Figure 51

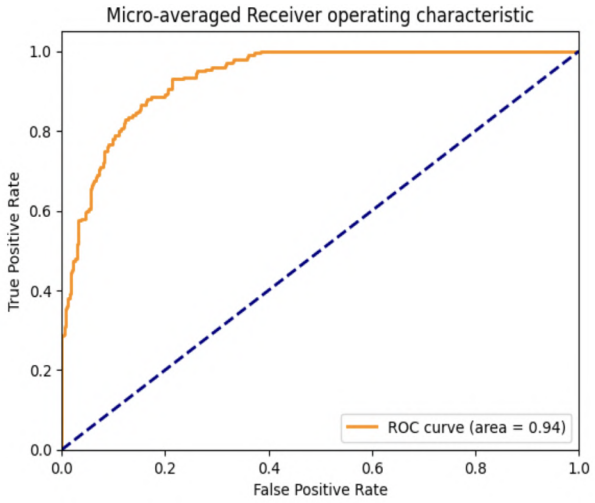
Classification Report for ResNet Model

	precision	recall	f1-score	support
dim	0.86	0.67	0.75	100
medium	0.70	0.88	0.78	92
bright	1.00	0.75	0.86	8
accuracy			0.77	200
macro avg	0.85	0.77	0.80	200
weighted avg	0.79	0.77	0.77	200

Figure 52 shows Receiver Operator Characteristic(ROC) curve and the Area under curve with the value of 0.94 for the ResNet Model.

Figure 52

ROC Curve and AUC Score of ResNet Model



When it comes to the DenseNet model, the accuracy that we have achieved is 64% which is lesser than the VGG16 and ResNet model. From the figure 53 below that shows the model's accuracy, we get the training accuracy of 86.88% whereas validation accuracy of 64%. When it comes to the model's loss we get training loss as 0.3114 whereas validation loss as 0.6941 as shown in the given figure 54.

The Confusion matrix for DenseNet Model in the given figure 55 below shows good performance of medium class as compared to dim and bright class. According to the confusion matrix, the model was only able to classify 49% of the data correctly, for the medium class it classifies 79% of the data correctly whereas for bright class it correctly classifies 75% of the data. From the classification report from the figure 56 below we can see that the precision is 0.86 whereas recall is 0.75 for the bright class.

FOR AUC_ROC curve as given in the figure 57 below we can see that the ROC curve is 0.93 for the DenseNet.

Figure 53

Training and validation accuracy for DenseNet Model

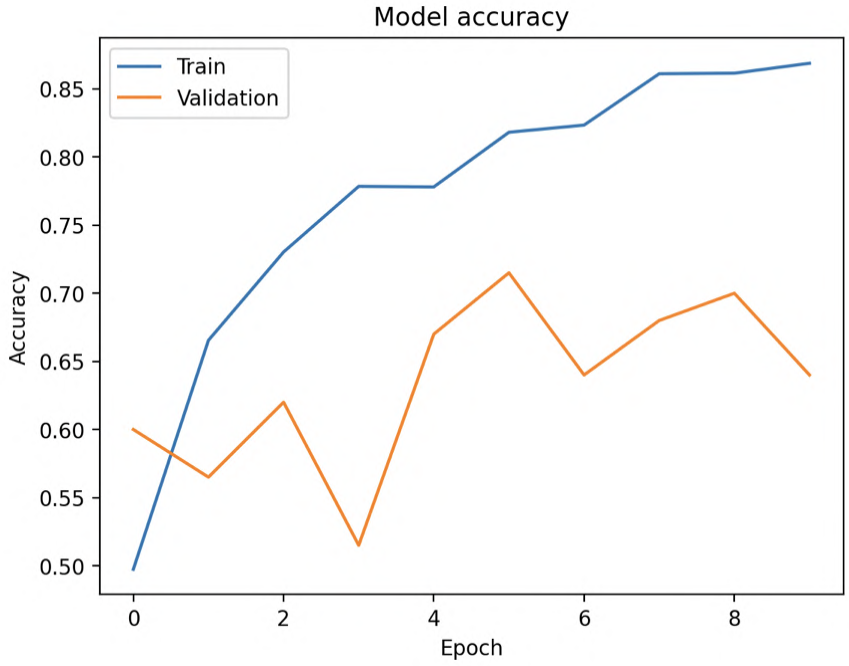


Figure 54

Training and validation loss for DenseNet Model

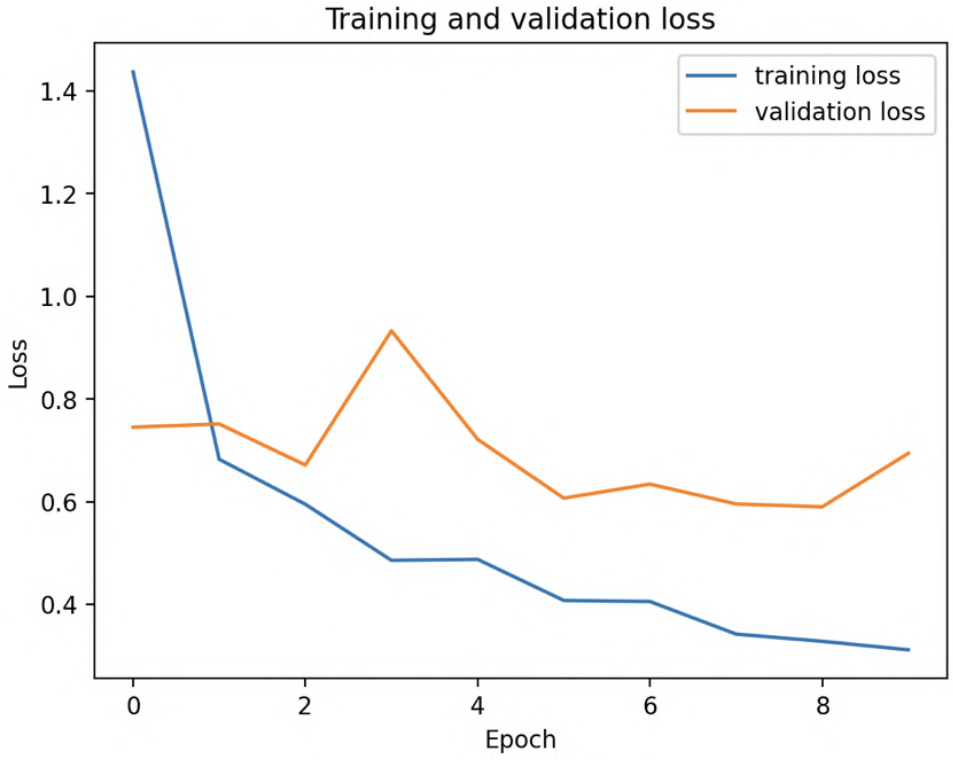


Figure 55

Confusion Matrix for DenseNet Model

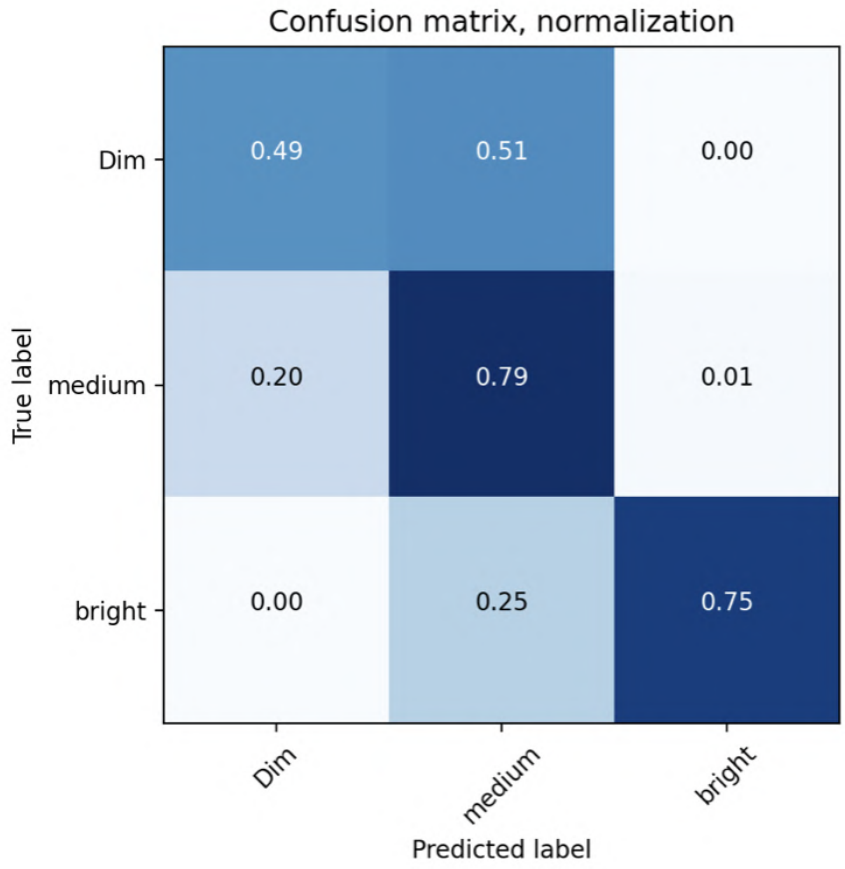


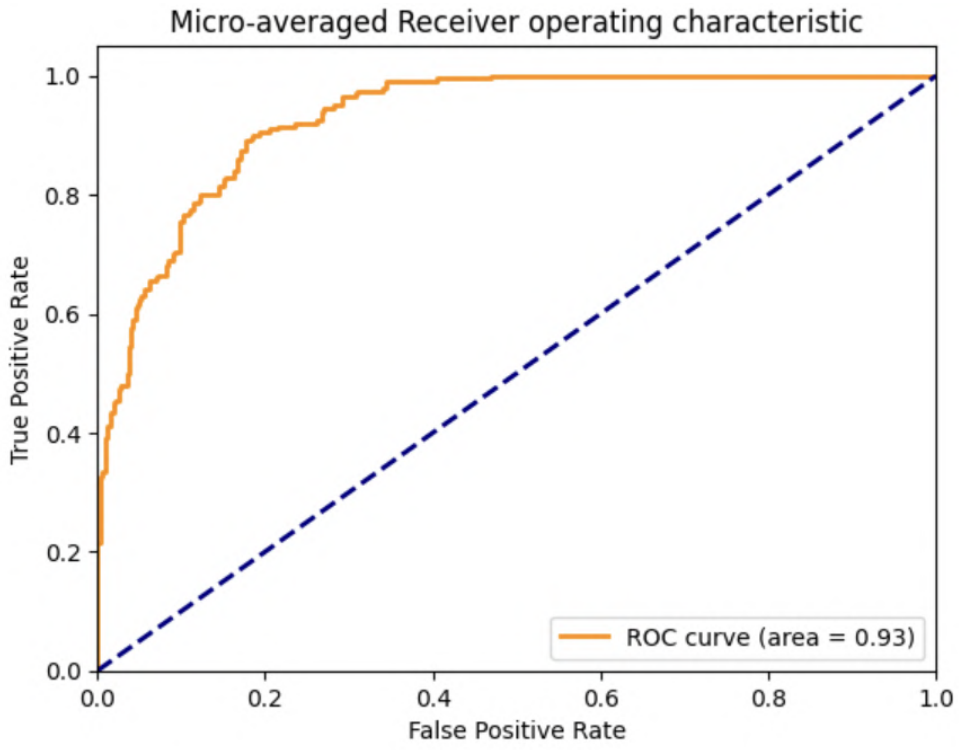
Figure 56

Classification Report for DenseNet Model

	precision	recall	f1-score	support
dim	0.73	0.49	0.59	100
medium	0.58	0.79	0.67	92
bright	0.86	0.75	0.80	8
accuracy			0.64	200
macro avg	0.72	0.68	0.69	200
weighted avg	0.67	0.64	0.63	200

Figure 57

ROC Curve and AUC Score of DenseNet Model



For the InceptionNet model, the accuracy of 62% has been achieved which is still not better than VGG16 Model. From the figure 58 below, we can see that the model's training accuracy is 89.35% whereas validation accuracy is 62%. The model's training loss in the figure 59 is 0.2746 whereas validation loss is 0.9516.

The Confusion matrix for the InceptionNet Model in the given figure 60 below shows very good performance of dim class as compared to medium and bright class. The model was able to classify 94% of the data as dim, whereas for medium class it was able to classify only 30% of the data and for bright class only 25% data was classified as bright by the model. The classification report in the given figure 61 shows precision of bright class as 1, recall as 0.25 and f1-score as 0.40.

The AUC-ROC curve for Inception Net Model in the given figure 62 below shows 0.82 as ROC curve area.

Figure 58

Training and validation accuracy for InceptionNet Model

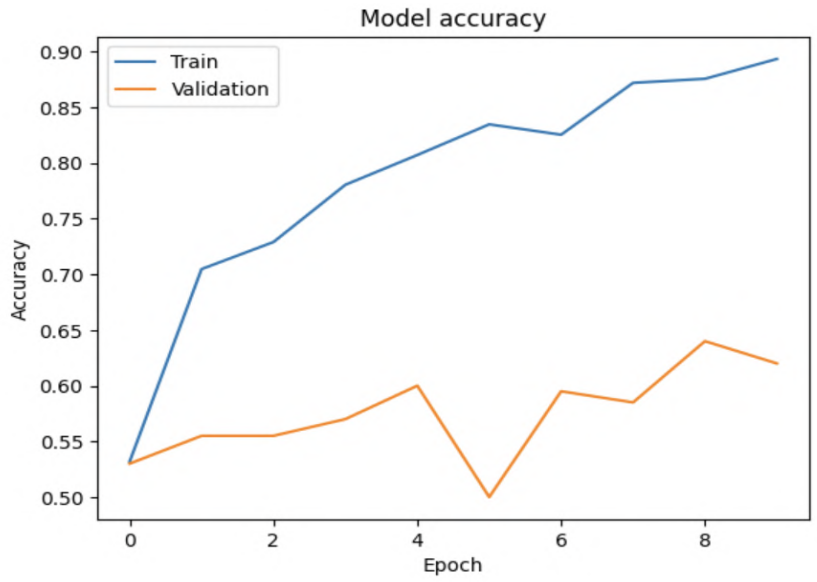


Figure 59

Training and validation loss for InceptionNet Model

Figure 60

Confusion Matrix for InceptionNet Model

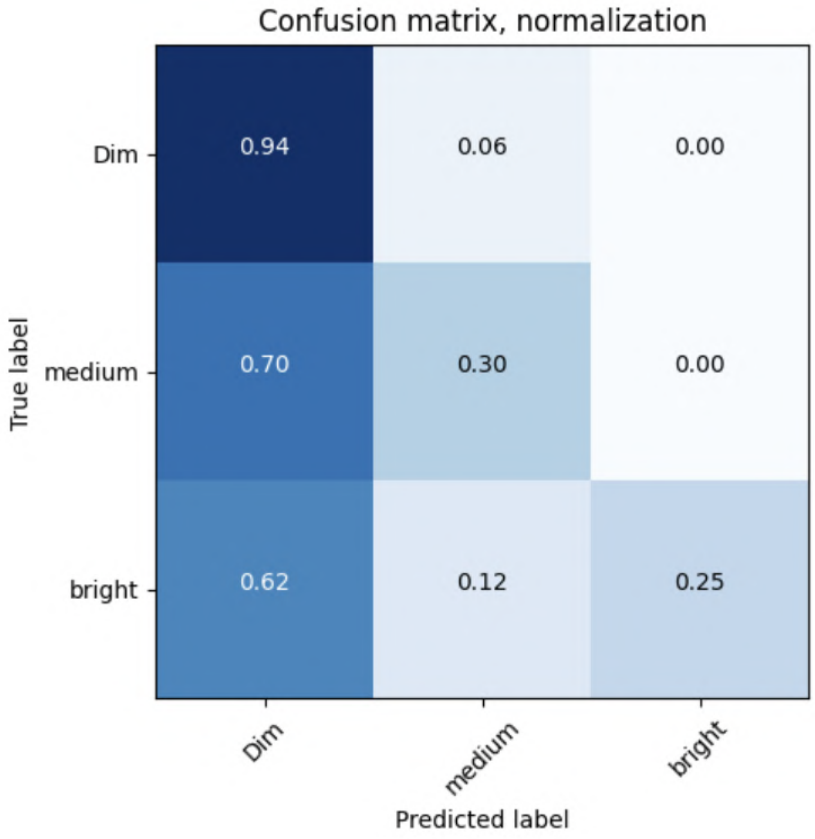


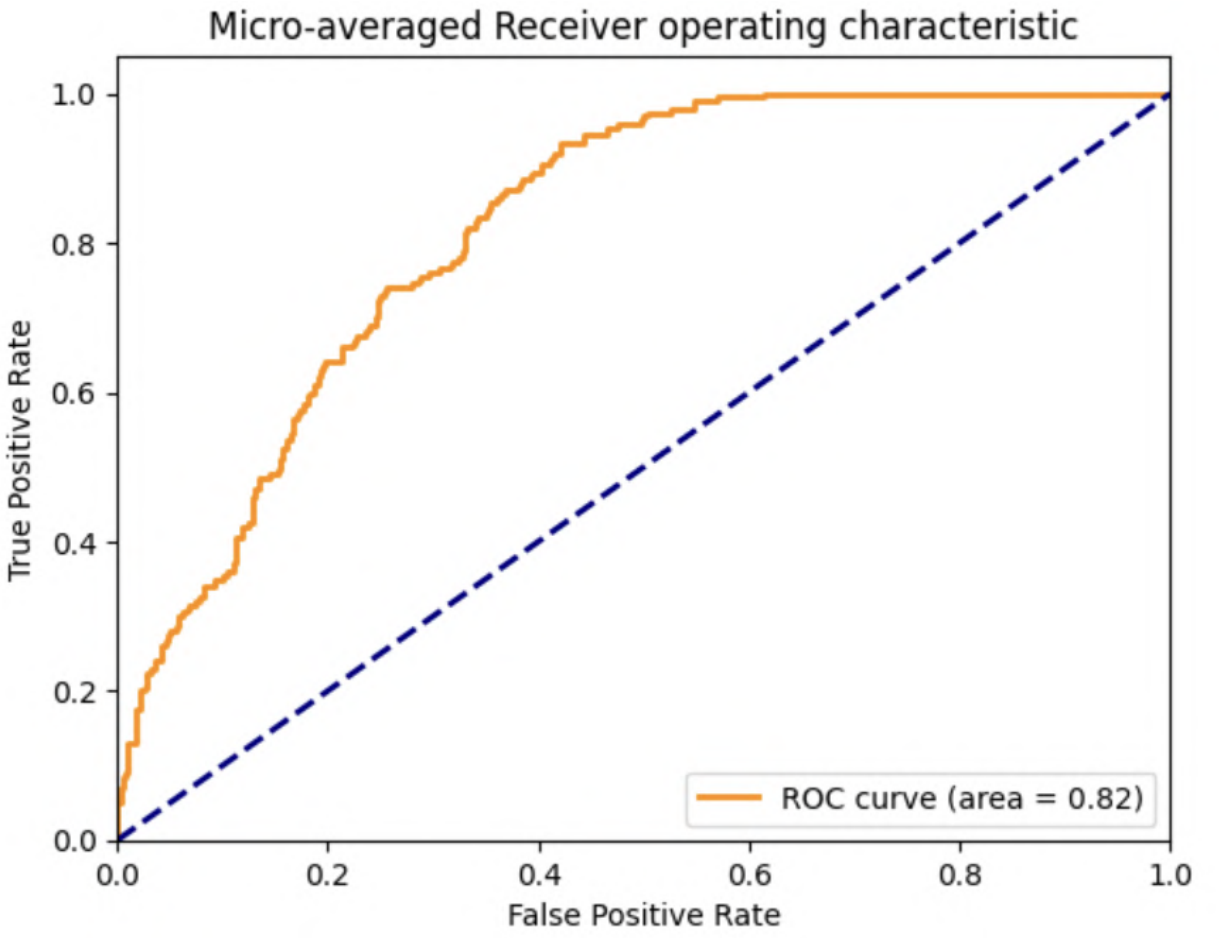
Figure 61

Classification report for InceptionNet Model

	precision	recall	f1-score	support
dim	0.58	0.94	0.71	100
medium	0.80	0.30	0.44	92
bright	1.00	0.25	0.40	8
accuracy			0.62	200
macro avg	0.79	0.50	0.52	200
weighted avg	0.70	0.62	0.58	200

Figure 62

ROC Curve and AUC Score of ResNet Model



The overall result comparison for all the models has been shown in table 7, where we can see the time taken by the model for training as well along with the accuracy and precision of each model. We observed that the ResNet- 50 model and the VGG-16 model with SE module performed almost the same in terms of accuracy and precision, but we also considered the time taken for training by each model while selecting the best model for deployment. So when we compare each model the maximum time taken by the ResNet-50 model was 1366.75 seconds and the time taken by the VGG-16 model with SE module was 1070.23 seconds which was better. So

to keep our application faster and get the results fasters, we decided to deploy the VGG-16 model with SE module integrated into our web application

Table 7

Result Comparison for All Models

Model Name	**Time Taken(sec)**	**Accuracy**	**Precision**
VGG - 16	1063.33	0.71	0.76
VGG-16 + SE Module	1070.23	0.77	0.77
ResNet - 50	1366.75	0.76	0.77
DenseNet	1215.12	0.64	0.67
InceptionNet 121	1255.93	0.60	0.63

Data Analytics Systems

System Requirements Analysis

System Boundary, Actors and Use Cases

The system boundary, actors and use cases for predicting poverty from satellite images using deep learning have been outlined in this section. The system boundary includes the images submitted by the users and the AI modules that will provide the analytical results to the users. A research organization or school that focuses on poverty and development, non-profit organizations that work to alleviate poverty, governments and policymakers, or a private sector business interested in funding projects to do so can all be considered important players in the system.The actor communicates with the system by uploading a satellite image of a specific place, as seen in figure 63. The pre-processing module will then handle the uploaded image, after which it will be given to the AI module. The user's uploaded picture will be processed by both AI modules as they work together, and the results or output produced by the modules will be displayed to the user. The poverty index module will generate a number that represents the index of poverty based on the picture, and the nightlight intensity module will predict the intensity of the lights at night for the image. Some of the use case for such a system has been given in the figure 63, and more information is given below:

Infrastructure planning. The web application that forecasts poverty from satellite images could be used by urban planners and municipal officials to help guide infrastructure planning decisions. Officials could enhance the quality of life for residents by prioritizing the development of infrastructure in areas with high poverty rates.

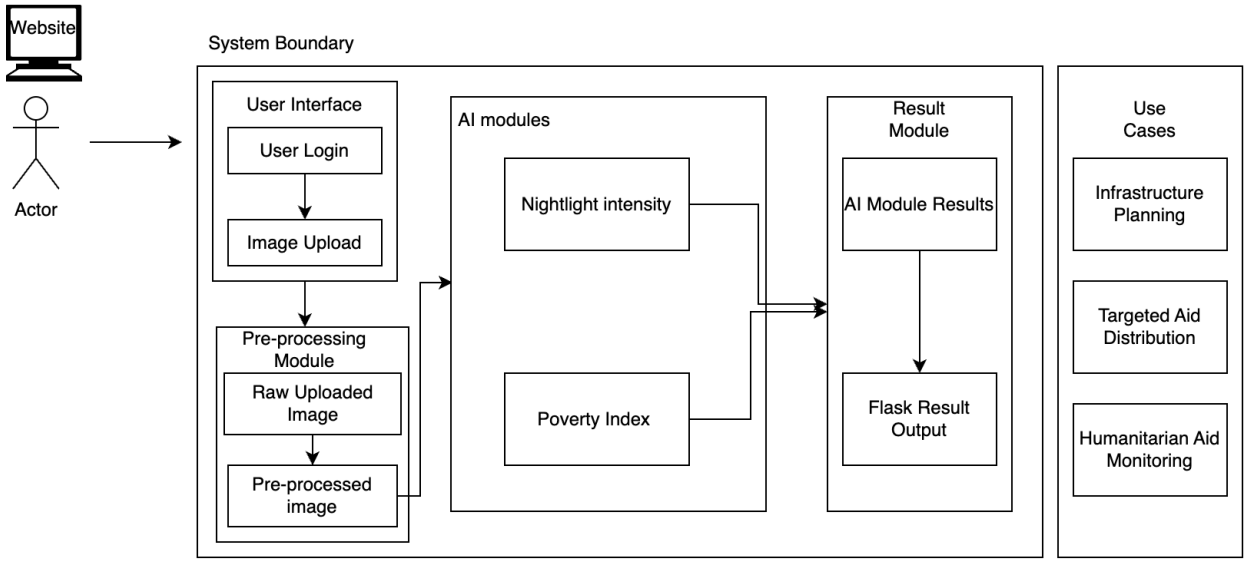
Targeted Aid Distribution. A web application that forecasts poverty based on satellite images could assist aid groups in more efficiently allocating their resources. Organizations could

make sure that their resources are going to the people who need them the most by identifying regions with high rates of poverty.

Humanitarian Aid Monitoring. The web application could be used by humanitarian groups to track the success of their aid initiatives. Aid agencies could assess the effectiveness of their initiatives and modify their programs by tracking changes in the rates of poverty over time.

Figure 63

Diagram of The System Boundary, Subsystem, Use Cases and The Actor on The System



Data Analytics and Machine Learning capabilities

The process of predicting poverty from satellite images using deep learning involves several data analytics and machine learning capabilities. Here are a few examples:

Image Processing. The first step is to process the satellite images using various image processing techniques. This includes tasks like image enhancement, noise reduction and feature extraction the goal is to create a clean and accurate image dataset that can be used for training machine learning models.

Feature Engineering. Once the images have been preprocessed the next step is to extract relevant features that can be used for training machine learning models. This involves identifying patterns and structures within the images that are associated with poverty; examples of features that can be extracted from satellite images include vegetation cover building density and road network density.

Machine Learning Models. Once the features have been extracted machine learning models can be trained to predict poverty levels there are several types of machine learning models that can be used for this task including convolutional neural networks (CNNs), random forests and support vector machines (SVMs).

Data Integration. Predicting poverty from satellite images often involves integrating data from multiple sources. This includes satellite imagery socioeconomic data and other geospatial data integrating data from multiple sources requires advanced data analytics and machine learning capabilities including data cleaning, data wrangling and data visualization.

System Design

System Architecture and Infrastructure

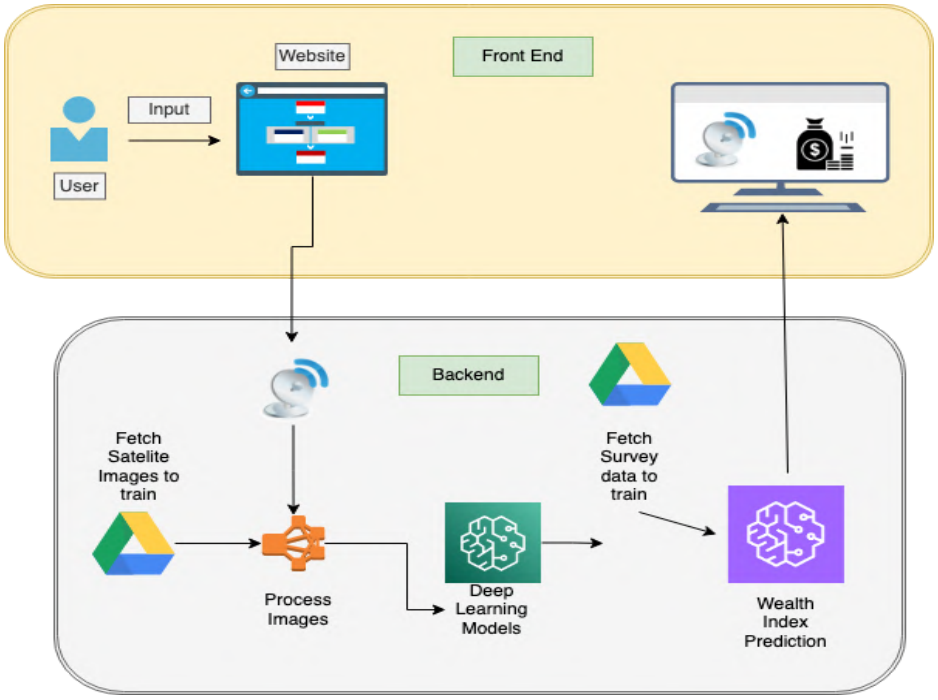
The frontend and backend system architectures are utilized to classify poverty levels from satellite images and the estimate of nighttime luminosity values are depicted. Figure 64 represents the system architecture. The solution makes use of Keras model and delivers it as a REST API through the Flask web platform. Users can upload satellite images to the frontend to anticipate the wealth index for such images. The backend system is split into two sections: data preprocessing and detection. The storage is set up to save all data on AWS until the account is deactivated.

Once a user submits a satellite image, the data is preprocessed and then sent through several detecting modules. To obtain an accurate representation of the wealth index, the model iterates over the image features. After that, the original image is displayed right next to the extracted sample, which clusters the agricultural region, construction area, roadways, and wasteland. The same satellite image may then be supplied to the frontend to anticipate the final index rate and produce the image's wealth index. The brightness and total wealth index rating are highlighted in this wealth categorization.

The images are initially processed by the AI module utilizing models such as VGG Net, Inception Net, ResNet, DenseNet, and Improved ResNet. Using the provided data, these models are trained. Once all of the images features have been discovered, a summary of the wealth index information is displayed to the user.

Figure 64

System Architecture

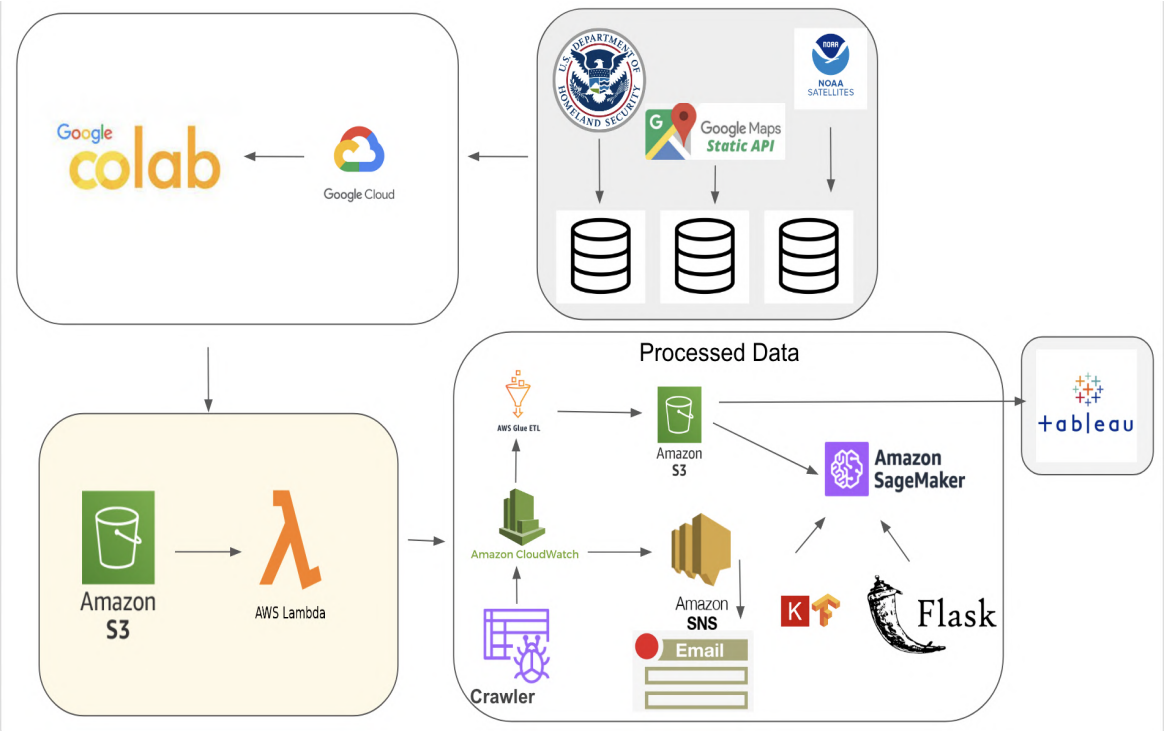


System Supporting Platforms and Cloud Environment

By leveraging deep learning on satellite images, the suggested application is intended to give significant insights on socioeconomic progress. The utilization of satellite images gives a unique viewpoint on land patterns, infrastructure, and other economic growth indicators that might be difficult to quantify using traditional methods. Figure 65 shows the application's backend infrastructure and how it will incorporate deep learning packages Tensorflow and OpenCV, allowing for accurate and rapid model creation. The Flask micro web framework will be used to construct the frontend, which provides a lightweight and flexible way to develop online applications. We can manage the integration of many models, as well as training image data for processing, using Amazon SageMaker, allowing us to develop more complicated and enhanced deep learning models. We can store and retrieve massive datasets for training data using Google Cloud, while simultaneously managing testing and user data easily. The interface of Colab and Google Cloud makes this process even easier by offering simple access to data and training tools. We'll move the code to Amazon once it's finished because Colab isn't available on the SageMaker environment. By moving the code to Amazon, we can use the SageMaker environment, which includes strong machine learning features such as built-in algorithms and automated model tweaking. Ultimately, the goal of this project is to provide a complete and effective tool for assessing socioeconomic growth, with a special emphasis on using deep learning and satellite data to deliver accurate and thorough insights.

Figure 65

Frameworks, Cloud Environment and Supporting Platforms.



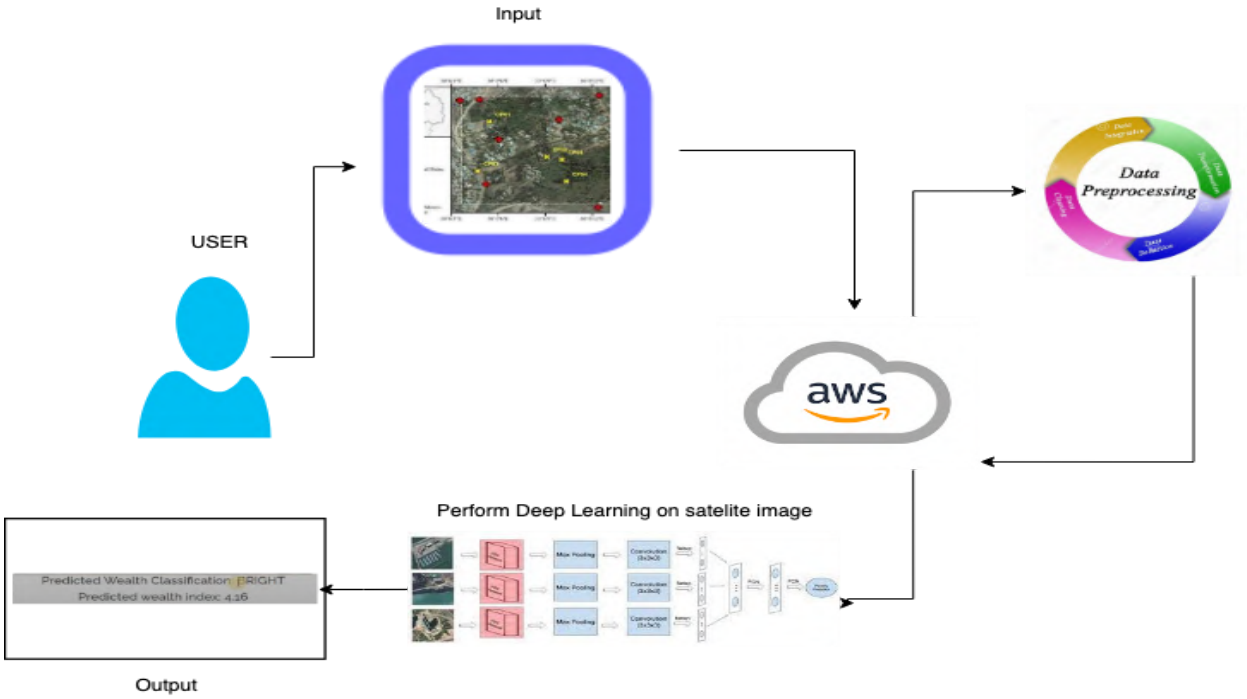
System Data Management Solution

Figure 66 depicts the data flow of the system. To begin the examination of socioeconomic growth, users must first choose a geographic image. Because the program does not need user registration, images are not saved on the front end. Users may only contribute images for analysis, not videos. The detected light brightness in the submitted images is then identified using detection modules, with the detection results given below. Data management includes gathering, organizing, storing, and utilizing data. Surveys and satellite data are used to collect data, which is subsequently stored in an S3 bucket. Google's Map Static API delivers daylight images in real-time image data in sizes ranging from 640 by 640 pixels. The Earth Observation Group supplies us with nighttime imaging data to assess poverty levels. Images

from both day and night, as well as text data, are kept in the S3 bucket, which is a cloud object designed to store data in a data lake.

Figure 66

Data flow chart from input to output

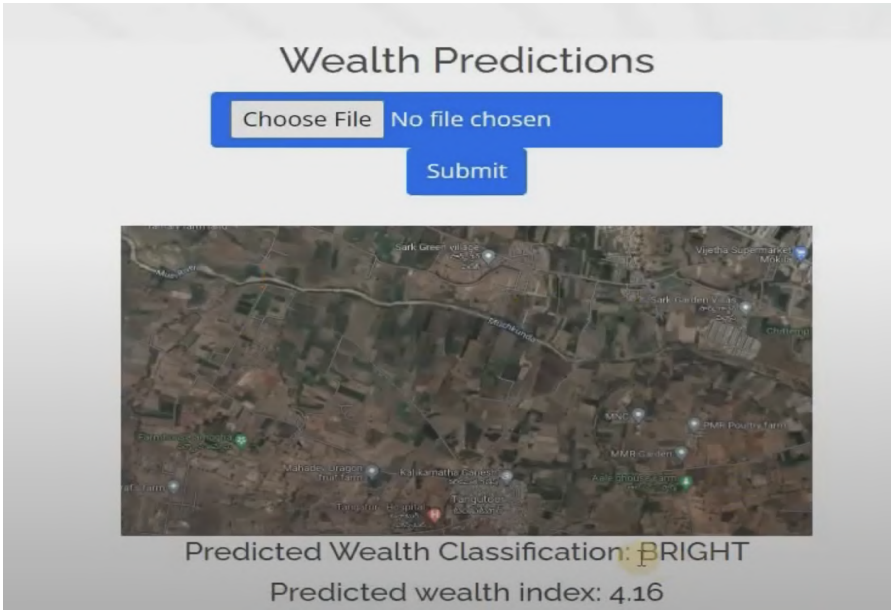


System User Interface

Figure 67 depicts interface examples for detecting poverty using satellite images. The graphic shows how users may submit a satellite image of their choosing by clicking the 'Select File' button. After successfully submitting the image, users may obtain results by clicking the 'Submit' button. The resultant information is displayed in the middle picture below, and users may compare the detection result image to the analysis image to determine if the satellite image is of good or low quality. Figure 67 shows that the wealth index is 4.16, indicating that the wealth is above average.

Figure 67

Examples of the interface for the examination of socioeconomic growth.



Intelligent Solution

AI and Machine Learning Models Development

This project is about detecting the poverty of a place by utilizing satellite images. As we are trying to use the images of a place in order to analyze the economic situation, we need to use a deep learning concept. The whole purpose of this project has multiple use cases. Financial departments of countries can make regular surveys about poverty and can have updated information and Governments also can keep a track of the economic health of a place to take necessary actions if needed. According to the results of the developed solution, governments can work on their budget to allocate the appropriate funds. Indirectly this project will help for many applications as a small module to get the information about the poverty of that place. For example companies can use this application to check the availability of affordable labor and low standard of living inorder to have least operational expenses on maintaining their companies.

For this project we have collected different data from different sources and combined them to get a meaningful dataset. We have collected the data from three different sources. We collected the survey data from the DHS website of Rwanda country, collected the night light image from NOAA website and collected images of Rwanda from google static map API. We collected all these data and did necessary cleaning and transformation steps and created a data pipeline.

We are using VGG Net, Inception Net, ResNet, DenseNet and Enhanced ResNet models to train with the prepared data. All the mentioned models are very popular in the deep learning field to work with images.Each model has its own advantages and disadvantages. We want to try all the models and see which model is more suitable for the data. That's why we will start with the simple CNN model that is VGG. This model is quite famous for its simplicity and accuracy in performance but it is slow to train and a bit large network with weights. The next alternative and better model is ResNet where we will get the advantage of avoiding the underfitting or overfitting of the model. We are using Dense which is different from ResNet and VGG. Inception Net is used to take advantage of kernel size. Apart from all these pre-designed models we are using the modified version of ResNet named Enhanced ResNet. In this model we changed the architecture of the model by integrating the SE module and Focal loss. We will train all these models and will come up with the best model and that model will be deployed into the application.

Implement Designed System

The proposed system needs to be designed in a way such that it should be user friendly for the proposed users and also should be capable of satisfying the targeted goals. We are developing a system which is a python application using the flask concept and hosted on a

server. This application needs to integrate with a good looking User Interface which will be easy for the users to use or to understand. One section of the application will be with all the details about the project, that is all the information about the sources we collected and flow of the project we followed and executed to develop the whole system. Another section consists of information about the observations that we found from the data collected and the sub section consists of all the deployed models in the portal with their performance metrics and all that stuff.

First we will collect all the data into data storage in AWS and then will start all the cleaning and transformation procedures on the data. We will use the ETL tool to do these all operations and load the data into the data warehouse. Then we will develop the model using deep learning to predict the wealth index of a place using the loaded data. After that we will develop the web pages for the website which are necessary using javascript, HTML and CSS also if required. Then we will integrate the saved models with these pages and will make a python web application hosted on some server and will deploy all the content into that website. After deploying all the content and models into the system then we will test the system within our team and also we will share this website with our friends to test and will ask for their feedback. We will take at least a week for this process. After receiving all the reviews from the users we will make all the necessary changes that we need to in the system and will finalize the system.

Implementing a designed system involves putting the plan into action and bringing the system to life. It is a critical step in the system development life cycle and requires careful execution to ensure that the system functions as intended. This process can involve several stages including coding testing and deployment and it requires the involvement of various stakeholders such as developers, testers and end users.

Input and Output Requirements, Supporting Systems and Solution APIs

According to the prototype of the project, we need different kinds of data in order to build the system. We need the data related to the wealth information of places and also we need night time images and day time images to map with the wealth information. Images are downloaded using the google static map API and also the NOAA for night time images. These images are cleaned using the data cleaning techniques and mapped with their respective wealth information. The developed or deployed system needs a satellite image of user choice to upload into the portal inorder to get the wealth index of that place.

Once the user uploads the image to the website then the image will be given to the trained model to estimate the wealth index of that particular place and returns the wealth index on the portal to display as the estimated wealth index. Apart from this, users can see the details of the project on the website including the data sources we collected and the data pipeline we made for this project and models we used for this project with their performance metrics.

This project provides the solution for the estimation of the wealth index of a place without any routine physical survey using expensive resources and by investing huge time on it. This approach saves a lot of time and also human effort which benefits the government or the institutions which are conducting these surveys to estimate the financial status of a place. Due to this we can get the updated information whenever we want without much investment and time. For this project we are using the google map static API for training the model. In the future we can integrate the API with this application such that we can trigger the service in the application to estimate the wealth index of all the places of a particular country. Then the model will return all the estimates of the wealth index of all places as a list to the user.

Google colab supports code development for model and data preparation. After completing the POC on the code we will load this code and data to AWS to train the model. Once the model is developed this model is integrated into the python application. Amazon AWS is supporting the deployment of the python application as a website. It is also supported to store the data and to host the website.

System Support Environment

Our project's purpose is to create a system that can anticipate the socioeconomic situation of diverse African areas, allowing policymakers to better focus social welfare and humanitarian aid, educate poor people, and analyze the results of policy efforts. Many environments were employed to assist the functioning and deployment of our product during its development. Jira has been used as a project management application to track the strategy, pipeline, and overall project development. AWS is used to store all the satellite images, survey data, and GPS data collected from various sources. Amazon Glue is used for data management and preprocessing. Amazon Redshift is used to display data visualization by connecting to Tableau. Google Collab is used to run all deep learning and machine learning models. On GitHub, a code repository is established, which also includes the project documentation. We will create a web application for deployment using React and the Flask API, and the next subtask will be to link the models with the completed portal. Upon integration, we'll monitor the application for any anomalies and, if necessary, offer maintenance.

Jira

Jira, a well-known project management tool, was used to organize our project, assign tasks, and check project status. Work breakdown structures (WBS), Gantt charts, and other

planning tools are only a handful of the many functions provided by Jira. Jira is a free project management tool that works well for us.

AWS

It was difficult to keep our data in a local system because it was in the form of satellite images, surveys, and GPS data. As a result, we employed amazon web services to solve this problem by constructing an s3 bucket, which is a cloud object used to store data in a data lake and importing the data into the s3 bucket and any amazon service would be able to access the data stored in s3.

AWS Glue

Amazon provides an ETL solution to efficiently handle data stored in several datastores. Glue also includes automatic Python scripts for removing duplicates and missing values from our data during preparation. The data is subsequently returned to the S3 bucket and is available for usage. It may also work with Amazon Redshift to handle and manage data.

Google Collab

While we are employing Deep Learning and Transfer Learning models to forecast poverty in our research, our local system is incapable of processing these very GPU heavy models. As a result, we are utilizing Google Colab to run these models.

Github

On GitHub, a repository is a storage directory that saves and maintains all of the project's files, which can include the created model, data processing code, and stored model, as well as detailed documentation such as the project report, presentation, and project requirements. It also allows you to evaluate code, make appropriate modifications and improvements, and monitor issues. The entire project team can work on the project concurrently.

System Evaluation and Visualization

Analysis of Model Execution and Evaluation Results

For our project we have decided to use the VGG as the final model for classification of the image based on the nightlight intensity. The model classifies a image into dim, medium or bright for the image uploaded by the user by using our developed user interface for the project. Using the VGG model we have achieved a good accuracy which is around 90%, a recall of 90% and an F1- Score of 89%. The model performs better when the epochs are more and a batch size of 32 as compared to lower number of epochs and batch size, and it will attain even better accuracy.

The results of our model have been evaluated using various metrics such as precision, recall and F1-score and are shown below. Figure 68 shows the confusion matrix obtained after training the VGG model and it shows after giving images of 3 different classes to the model it correctly predicts 67% for class 1(dim), 95% for class 2(medium) and 50% for class 3(bright) and figure 69 shows the training accuracy and loss for the model as it shows sudden drop in the training loss after 1 epoch and continues to drop as number of epoch increases whereas accuracy increases as the number of epochs increases. A classification report was also used as shown in figure 70 which shows the model accuracy of 79% in predicting poverty. It also shows the number of images we have passed to the model which is 200 images in total including 100 dim class images, 92 medium class and 8 bright class images. It also indicates macro average precision, recall and f1 score as 88% , 71% and 75% whereas for weighted average it shows 83% as precision, 79% as recall and f1 score as 79%.

Figure 68

Confusion Matrix from model training

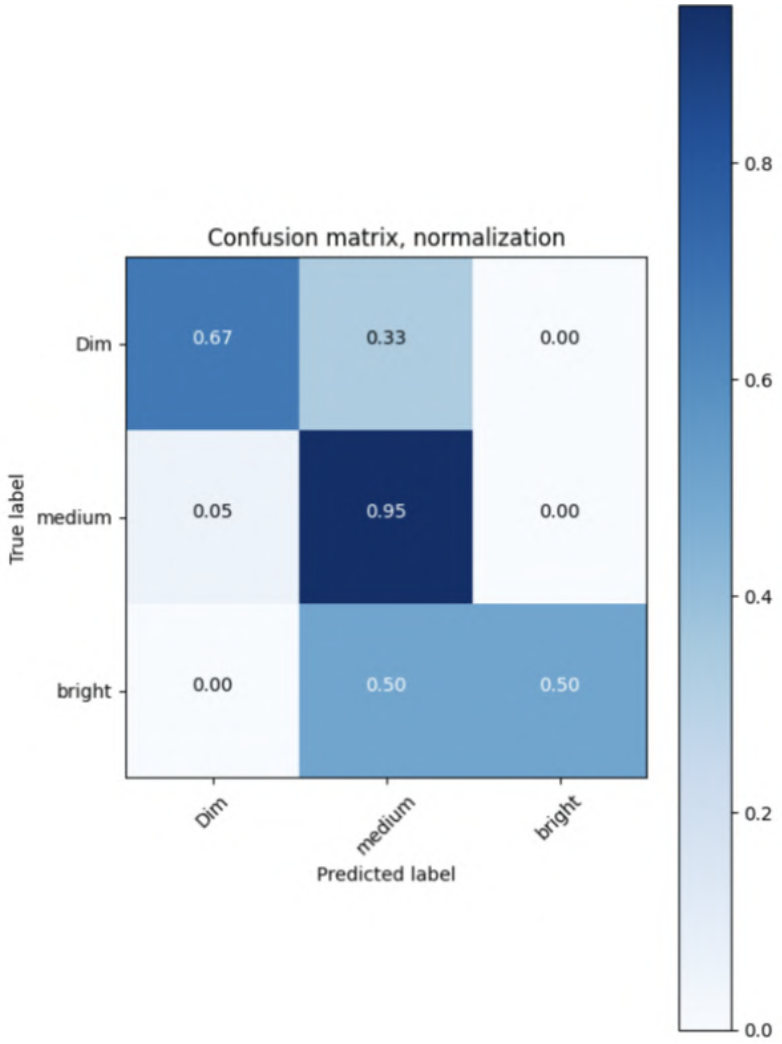


Figure 69

Curve representing training accuracy versus loss for each iteration

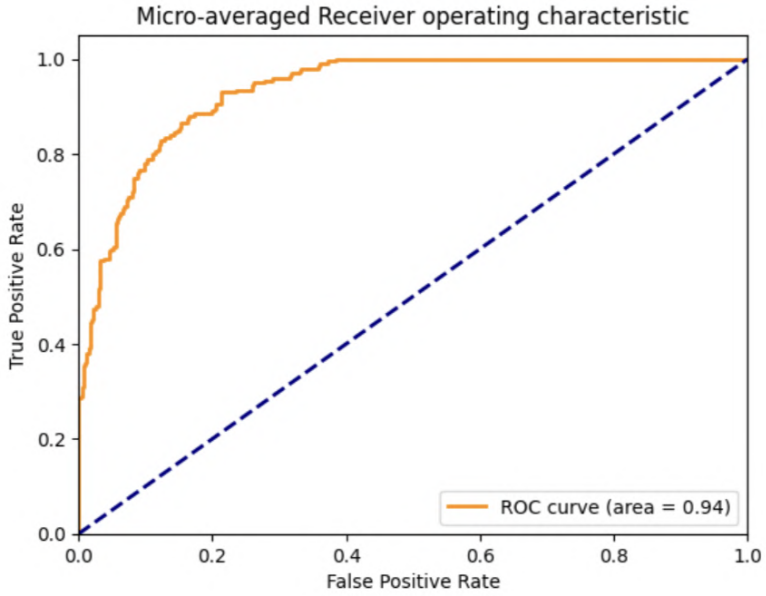
Figure 70

Classification Report from model training

	precision	recall	f1-score	support
dim	0.93	0.67	0.78	100
medium	0.70	0.95	0.81	92
bright	1.00	0.50	0.67	8
accuracy			0.79	200
macro avg	0.88	0.71	0.75	200
weighted avg	0.83	0.79	0.79	200

Figure 71

ROC-AUC curve



A ROC curve is created by plotting the true positive rate versus the false positive rate for the model at different thresholds. The orange color curve represents the area covered under the curve, we can see that the area covered for our model is 0.94 which is a decent amount.Thus it shows the model has the ability to classify the image in different classes.

Figure 72

Test accuracy and loss

```
7/7 [==============================] - 2s 261ms/step - loss: 0.4753 - accuracy: 0.7900
Test loss: 0.4753
Test accuracy: 0.7900
```

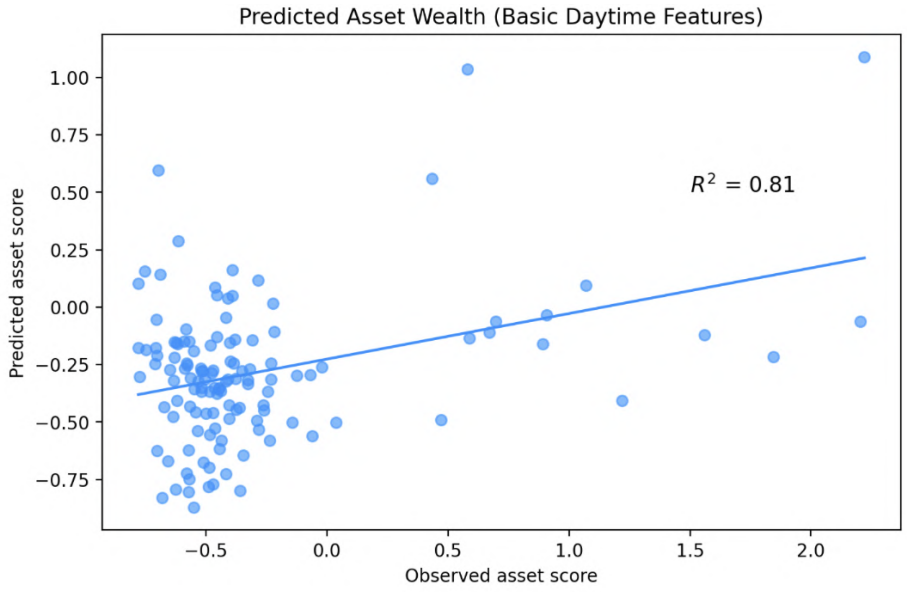
Figure 72 shows the loss and accuracy for our model on the test dataset. The total time taken for prediction of test data by our model was 261ms, and the loss reported was 0.4753 and the testing accuracy was 79%.

Figure 73 shows a graph of predicted asset score versus the observed asset score from our regression model. R2 score shows the relationship between two variables. A low score means

that the variables are not correlated and the model is unable to determine a relation between the variables.Thus a high R2 score shows that the model is able to predict a wealth index which is correlated with the given wealth index.

Figure 73

R2 score for the model for predicting wealth index



Achievements and Constraints

Achievements

We have successfully created a deep learning model for forecasting poverty using satellite images after performing in-depth study and analysis about the topic. To gather the relevant data required to build a useful system, the project needs a solid foundation of technical and literary surveys of how to predict poverty and can make predictions on it. Convolutional neural networks were especially used in our method of creating the deep learning models in order to provide precise predictions for poverty level.

We used satellite images to extract important characteristics of poverty, such as the standard of housing, the availability of power and water, and the state of the local environment.

We were successful in forecasting the levels of poverty in various places using deep learning algorithms.

Our algorithm beat earlier methods for predicting poverty, which frequently depended on laborious surveys and manual data collection. We were able to classify and identify poverty with high accuracy percentages, which might have a big influence on policy and resource allocation for a particular location depending on needs and level of poverty.

To make it simple to retrieve the predictions from our model, we also created a user-friendly interface web application where users can enter significant information on our website, which would serve as a conduit between them and the predicting model, and promptly obtain predictions of poverty levels. Our project had a low development cost since we used free software such as AWS and pre-existing satellite images for our project.

Ultimately, our approach has the potential to significantly influence how resources are allocated and predictions of poverty are made. We were able to create a very accurate and effective model by employing deep learning techniques, and this model may offer insightful information on the extent of poverty in various places.

Constraints

We ran across numerous issues that needed to be resolved as we developed the project to implement deep learning models to forecast poverty using satellite images. The data collection and data preparation process was the main challenge we faced as we made access requests to use this kind of data in our project. After getting access and finding relevant enough high-quality satellite images for the area of interest was a challenging part, and identifying those images took a lot of immense work too. Also, it took a lot of investigation and testing to extract the necessary data, including the vegetation indices.

Due to the lack of labeled data indicating poverty levels on the satellite images itself, which results in the absence of ground truth affects the supervised learning approach and it requires more domain expertise to label the images and connect it to the ground truth by using the survey data.

Second, choosing the proper deep learning models for this task presented was another difficulty. Before finding a good model that produced appropriate results, we tried with a number of alternative models and parameters. Furthermore, computationally demanding, the training of the models requires access to high-performance computing resources.

Thirdly, it was difficult to incorporate the deep learning models into a web-based interface. We had to guarantee that the web interface was responsive and simple to use, as well as transform the trained models into a format that could be readily put into it.

Notwithstanding these obstacles, the project's accomplishments were noteworthy. We were successful in accurately forecasting the prevalence of poverty by utilizing deep learning techniques. To help individuals live in better conditions, this information may be used to pinpoint regions that need more assistance or actions.

There are several restrictions attached to this method, though. This project may not be suitable to other locations with distinct features since it was created for a particular region. Also, the quality of the input data may affect how accurate the predictions are, and the deep learning models may need to be retrained using fresh data to keep up accuracy.

Also budget constraints also affected the project, as a limited budget and resources affected the scale to which the data was collected, also the models developed and evaluation efforts made by us as the online platforms such as AWS are very expensive.

Ultimately, a significant amount of time, effort, and resources were needed to produce this project. The findings do show that deep learning methods may be used to forecast poverty levels using satellite photos.

System Quality Evaluation of Model Functions and Performance

This section is about evaluating the system quality in terms of model correctness and runtime performance of them. Correctness of models is about solving the targeted problem in the project. In our project we are talking about the prediction of the wealth index of a city using its satellite image. Our developed system is capable of that. Our model prediction is better than the random guessing model which is a basic criteria for any kind of model development. Now runtime performance of the models is considering the training time and testing time of a model. These metrics play an important role in the real time deployment of the modules. For suppose if you are integrating this model with some other module which they are using in live such that model decisions/predictions will be implemented immediately within seconds. Then the model needs to predict very fast within some milliseconds. But in our case we don’t need any such requirements as this model will not be used in any kind of live module with such a requirement.

In this project we applied many models for the collected data. Among all those models, the best model was integrated with the developed system. Now we are going to evaluate the model with respective system requirements. VGG16 with SE module is the model performed best among all other models in this project. It took 12 mins time to extract the features from the images and 5 min time to train the model. This model is able to predict the wealth index cluster within 1 second. Which is fast and accurate. The overall performance of the model in terms of accuracy is 77% which is good to consider and proceed further. Apart from this model, remaining models' performance are also mentioned in the below table 8. As shown in the table

there is not much difference in the training time and testing between the VGG-16 and VGG-16 with SE module but accuracy differs by 6%. ResNet model is taking a longer time than all other models which is 22 mins including extracting features and training the model.

Table 8

Model Performance

S. No.	**Model**	**Device Specification**	**Training Time**	**Testing Time**
1.	ResNet	Google Colab GPUs : T4 CPUs: 2 x vCPU RAM: 32 GB	1366.75	243 milli sec
2.	DenseNet	Google Colab GPUs : T4 CPUs: 2 x vCPU RAM: 32 GB	1215.12	121 milli sec
3.	VGG-16	Google Colab GPUs : T4 CPUs: 2 x vCPU RAM: 32 GB	1063.33	46 milli sec
4.	Inception	Google Colab GPUs : T4 CPUs: 2 x vCPU	1255.93	248 milli sec

S. No.	Model	Device Specification	Training Time	Testing Time
		RAM: 32 GB		
5.	VGG-16 with SE Module	Google Colab GPUs : T4 CPUs: 2 x vCPU RAM: 32 GB	1070.23	50 milli sec

System Visualization

We collected different types of data from different sources. The collected data can be grouped into her three categories: survey data day and night images. The proposed system aims to estimate a place's wealth index using satellite images. This section will present a sequence pattern of the system from the beginning to the end by providing essential pictorial representations. Initially, we will discuss the data used in this project, which includes satellite images obtained from Google through the Google Maps Static API. The figures depict notable differences between the images. One image exhibits diverse structures such as roads, buildings, airports, flights, and patches of agricultural land, while the other image predominantly displays forest vegetation. Even without applying any financial analysis techniques, we can conclude that the place depicted in figure 74 is more developed than figure 75, indicating that the former's wealth index will be greater than the latter's.

To gain further insights into the level of development of different regions, we not only rely on daytime images but also on nighttime images. In particular, the main feature of nighttime images is the light intensity, as developed places tend to have more lighting of various kinds. For

instance, we can observe the nighttime image of Las Vegas, a sub-urban city in California, in figure 76. This image is dominated by city light intensity and provides valuable information about the city's level of development. Similarly, if you look at the satellite image of the united states in figure 77, it shows some parts of the image have maximum light intensity and other parts do not have full light intensity the light intensity in nighttime images is a useful indicator that allows us to determine the level of development of a city or place by analyzing these images. We gain valuable insight into the state of infrastructure development, economic activity and other factors that contribute to the wealth index of a place.

Figure 74

Example of highly developed area

Figure 75

Example of under developed area

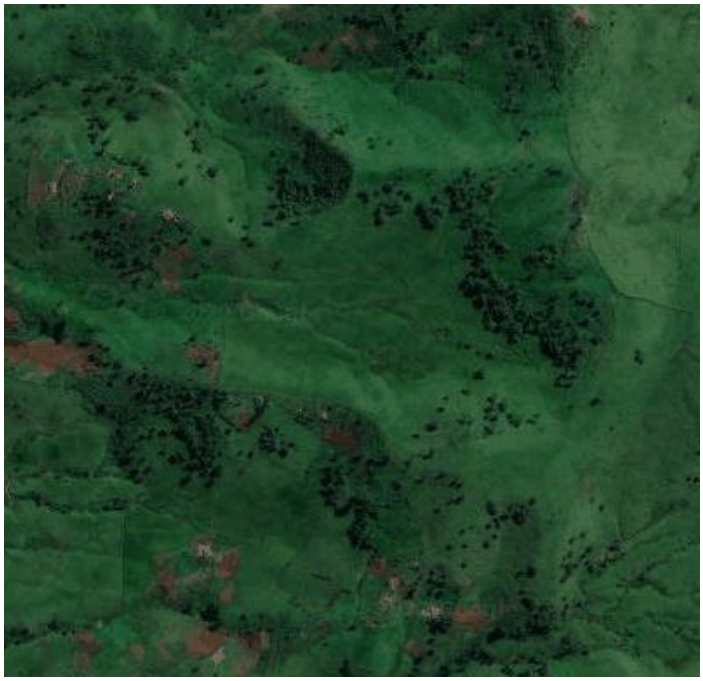


Figure 76

Figure showing night light image of las vegas area

Figure 77

Night light image example from NOAA

Now, we will apply a few visualizations on the collected data and analyze them. The figure 78 shown below is a scatter graph between the cluster and wealth index. Collected data is about the Rwanda country which consists of around 500 clusters. Each cluster's wealth index is represented with the corresponding cluster number in the graph. If you observe the data, the majority of the data lies below the value zero and remaining are majorly distributed between zero and two. We can find a few clusters with the wealth index more than the value two. The next figure 79 shows the classification of the images into three major classes i.e., dim, medium and bright images. In total images, major images i.e., around 23K are classified as dim images and the remaining few are medium and bright images. Figure 80 shows the graph between the average night light luminosity and average cluster wealth. We can observe that the average cluster wealth index is increasing with the average light luminosity.

Figure 78

Scatter plot for average wealth index for each cluster

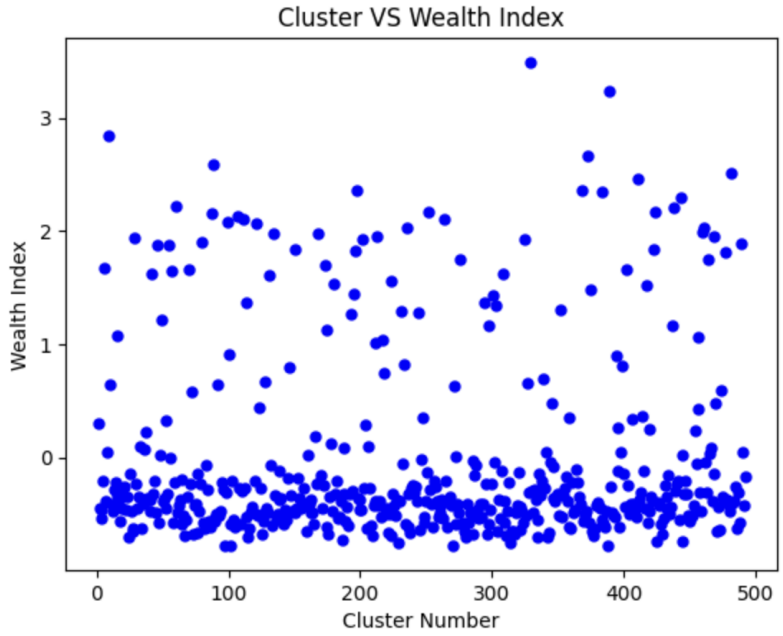


Figure 79

Bar plot representing collected images from each class

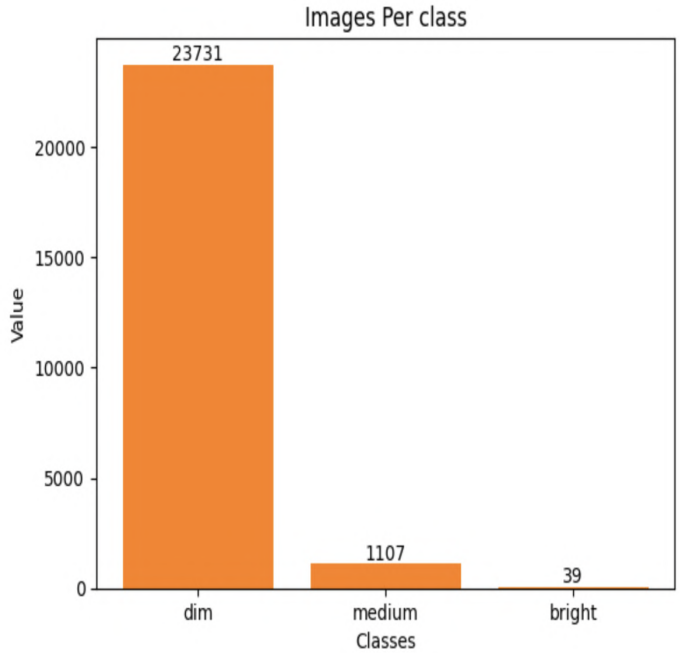
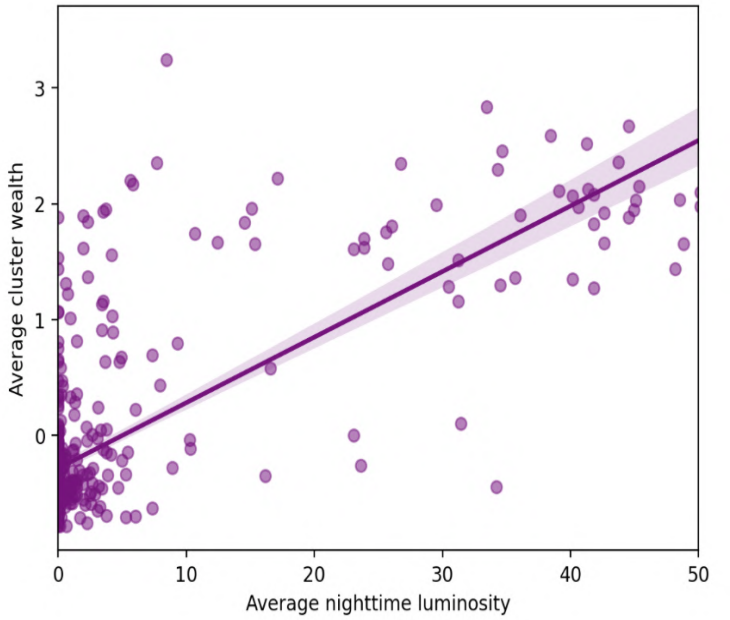


Figure 80

Scatter plot for nighttime luminosity versus average cluster wealth



As we discussed earlier, we integrated the best model with the application. Now we are representing a few results from the predictions of the model. The first figure 56 shows the wealth index as 4.16 with the bright classification. We can observe in the image that it is full of buildings, roads, water bodies and different structures. If you observe the other two figures 82 and 83, the wealth index is lower than figure 81. For the second figure, due to the presence of helicopters the wealth index increased whereas in the third figure there are not that many buildings and structures, which affected the predicted wealth index.

Figure 81

Example of UI predicting image from bright class

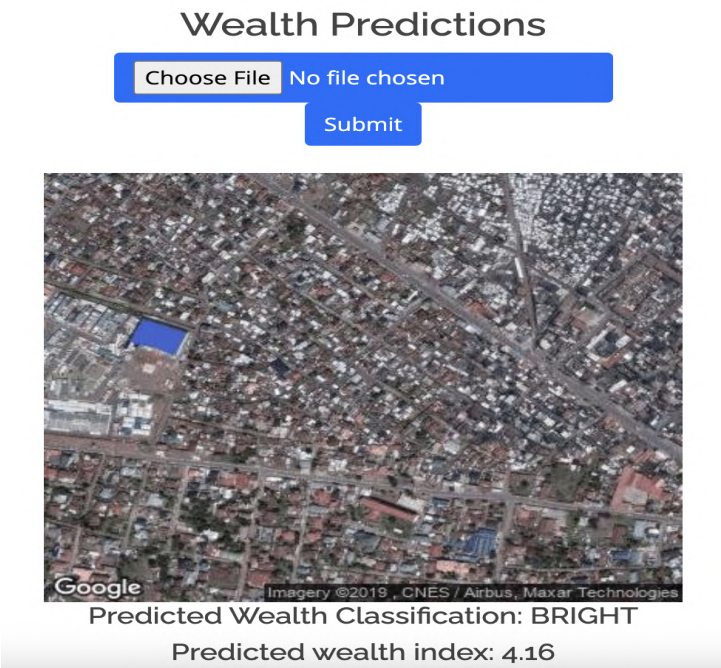


Figure 82

Example of UI predicting image from medium class

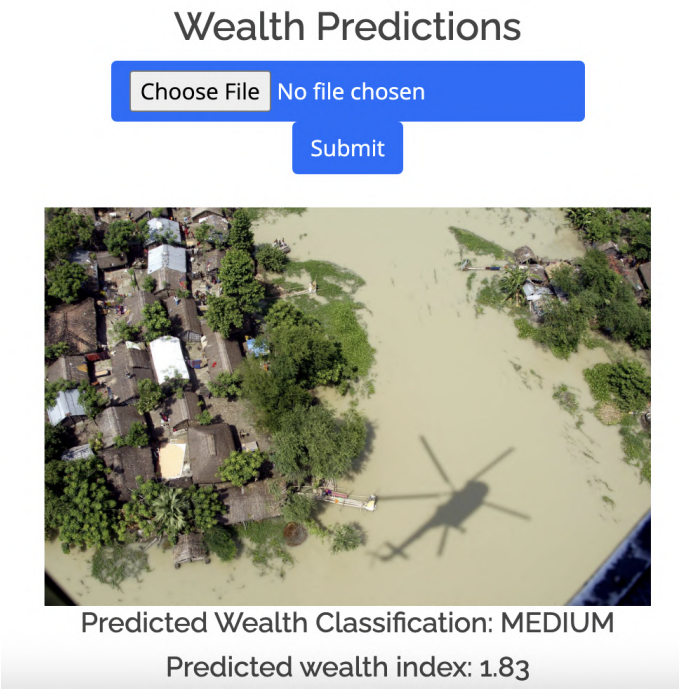
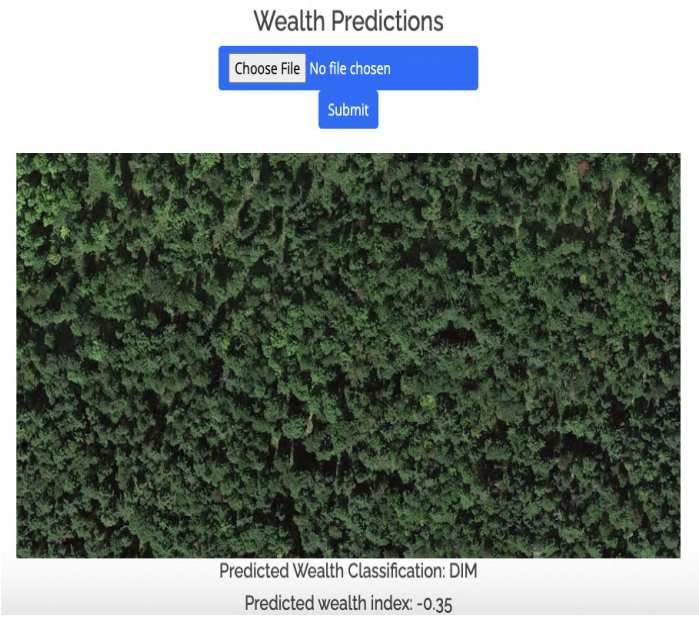


Figure 83

Example of UI predicting image from dim class



Conclusion

Summary

In this project we developed a system to track the socio-economic status of a place with the satellite image of that place. This system consists of many components such as backend and frontend and pipeline for data processing. Application was developed on Flask concept in python language. In our use case we deployed this web application on the cloud infrastructure whereas in the real world we can install on private servers of clients which gives more security and safety for the data consuming and data generating. Organizations/Users can use this in multiple use cases. One is predicting the wealth index of a place by uploading satellite images to the web portal. Another case is like initiating the process to assess the economic status of the whole country by uploading all the pictures of places covering the country. Now we are summarizing the whole project as below.

For this project we have collected different kinds/types of data from multiple sources and combined them to train the model. We have collected survey data from DHS website for the country called Rwanda for the financial year of 2014-2015. We applied for the access of this data by submitting project proposals to them. Parallelly we have collected night light images of the world from the NOAA website with high resolution quality. Image is with the tagged coordinates which is flexible to process those images. We need day time satellite images of those places, for that we need to have boundary shape files of those clusters. Then we downloaded the boundary file of rwanda and then with the help of that we collected day time images from google static map api. This is the whole process we followed for collecting the required data for our project.

We followed the CRISP DM methodology to develop the project. We have used pert chart, gantt chart to schedule the task and to maintain the deadlines for those. We have used JIRA

software as a project management tool to assign the tasks, to track the progress of those tasks and etc.

After collecting these datasets, we opted for cloud amazon AWS for implementing the project. We deployed all the collected data on S3 bucket in the amazon and started making a pipeline to make the data ready for the model training. Each data went through many preprocessing steps to prepare the data. Survey data was filtered and integrated with the cluster information and deployed into S3 bucket. Features are extracted from night light images and merged with the wealth index of the cluster. Parallelly features are extracted from day time images and integrated with wealth index and cluster information. So, by the end of preprocessing we prepared three datasets and prepared into three subsets in a ratio of 70:10:20 for training, validation and testing respectively. As part of the pipeline we used amazon sage maker for preprocessing the data and EC2 instance for hosting web applications.

We went through so many research papers, articles and blogs to select the models for this project. As we are using images we need to consider Deep learning models. There are so many popular CNN architectures for images. We explored many CNN architectures and shortlisted a few of them according to the suitability and flexibility of them to our considered targeted problem. We considered ResNet, VGG16, Inception, and DenseNet for model development. As an innovation we implemented a SE module and integrated it with all the models for improvement. VGG-16 outperformed all other models and the SE module increased the performance by 6% accuracy. We considered accuracy, precision, recall as evolution metrics for our models. We considered simple evaluation metrics for this project.

The best model from our implemented models is VGG-16 with SE module with 77% accuracy. Apart from that we also implemented a linear regression model to predict the wealth

index of that particular place. So, our system can predict the wealth index and also classify the uploaded picture.

Benefits and Shortcoming

Benefits

The Proposed solution in this project for the targeted problem was beneficial in many ways than the traditional and existing solutions. This approach saves lots of time, human effort and expenditure on conducting those surveys. We can download the latest satellite image of a place and can see the economic status. This approach will allow us to cover the places from small scale to large scale within a region in order to analyze the trend or pattern to understand the growth. We can even assess any place on the globe without having physical access to that region.

Shortcomings

This project is using satellite images of places which are very confidential sometimes for some locations. When a government or organization is using this application then we will access the live images of locations for better prediction. Those images are very important in the subject of data security, data safety and consent. We trained our model only using Rwanda, which is one country out of 196 countries in the world. So, this model may not be good for other places outside of rwanda. We considered images with 400X400 pixels due to memory limitations which can affect the quality of the data extracted from those images. We got night light time images of the world of the year 2017, 2014-2015 survey data and latest available day time satellite images from google. If you observe here clearly, except the day time, remaining are from different time periods which may not be good to integrate and train the model.

Potential System and Model Applications

This project has many various potential applications in the real world. The most important application of this project is for governments. They can use this application to track the wealth index of places. Apart from governments, any authenticated organizations, those who are interested in conducting surveys to assess the socio-economic status of that place they can use this application. Governments can assess the change in wealth index regularly by initiating the process to assess them in regular periods of time. Another important application is extracting the features from satellite images. This can be used for analyzing the growth of a city after making any policy implementation or any impacting actions on a place. This application can be integrated with any other related applications such as agriculture yield, nutrition assessment of a place, e-commerce website for analyzing sales impacting factors, etc.

This application can be integrated with Real Estate companies to assist them in their future projects to select places for development. This application can also serve international institutions to analyze the growth for maintaining investments relations etc. with the other countries. These are some potential applications of this project. There are still more applications where we can use this application as a sub-module to use the inputs coming from that.

Experience and Lessons Learned

This project has been a valuable learning experience for our team from a technical point, we have gained a lot of knowledge and skills from taking this project course, we successfully built a full scale pipeline in AWS and we gained experience in deploying applications in AWS cloud 9. Our deep learning skills have increased as we learned various deep learning models and our data engineering skills have improved throughout the project. We gained experience working with api's for satellite imagery which has increased our technical capabilities this project

provided us with information into the wealth status of different regions worldwide and showed us areas of the world that are struggling with poverty, it has deepened our understanding of the global poverty crisis and its impact on different communities, this project has increased our technical skills and expanded our knowledge in deep learning. Our knowledge and understanding of the poverty and wealth gap across various regions has also increased, this project has been a great learning experience for our growth and has taught us practical skills and knowledge that we can apply to future projects.

Recommendations For Future Work

Based on our experience with the current project we have a few recommendations for future works and extensions first its important to prioritize improving the accuracy of the wealth evaluation model we can achieve this by using advanced deep learning techniques and incorporating additional data sources and implementing other validation procedures. This will ensure more precise results second its important to consider additional socioeconomic factors like education levels healthcare access and infrastructure quality these factors provide a better understanding of wealth and poverty dynamics by analyzing them we can gain valuable insights furthermore keeping a close eye on trends over time allows us to better understand how policies are working and make smarter decisions

This valuable information gives us the tools to provide policymakers with targeted strategies to tackle poverty effectively by using the models outputs to make recommendations we can help shape policies to make a difference in peoples lives we must also remember to uphold ethical standards protect privacy and collaborate closely with stakeholders their valuable insights and involvement are crucial to the projects success moreover we should prioritize scalability replicability and open knowledge sharing these factors are key to driving progress and enabling

the implementation of similar projects in diverse contexts together we can make a meaningful impact and create a better future for everyone involved.

Contributions and Impacts on Society

The project has the potential to make a real impact in diverse communities it focuses on including everyone by considering different factors like income and background and tailoring approaches to fit each community's needs the project findings help guide policies and actions to reduce poverty and create fairer opportunities for all in terms of education. The project helps us understand how education affects how much money people have and this information can be used to make sure that everyone has equal chances to succeed on a social level. The project raises awareness, starts important conversations and involves communities in finding solutions to poverty and inequality, this brings people together builds support networks and makes our society stronger. Additionally the project insights contribute to global discussions where we can learn from other cultures and share ideas to fight poverty and improve well being worldwide together we can make a real difference and create a better future for everyone.

References

Alsharkawi, A., Al-Fetyani, M., Dawas, M., Saadeh, H., & Alyaman, M. (2021). Poverty classification using machine learning: The case of jordan. *Sustainability*, *13*(3), 1412.

Aly, S., Alfonse, M., & Salem, A. B. M. (2022). Bankruptcy Prediction Using Artificial Intelligence Techniques: A Survey. In *Digital Transformation Technology* (pp. 335-360). Springer, Singapore.

Attention Required! | Cloudflare. (n.d.). https://datarade.ai/

Browne, C., Matteson, D. S., McBride, L., Hu, L., Liu, Y., Sun, Y., ... & Barrett, C. B. (2021). Multivariate Random Forest prediction of poverty and malnutrition prevalence. *PloS one*, *16*(9), e0255519.

Data Page. (n.d.). https://nsdi-rla.hub.arcgis.com/

Dharani J. (2020) *All About ML — Part 1: Detailed explanation of Linear Regression* https://medium.com/all-about-ml/linear-regression-d41a6a5dcab6

DHS Recode Manual (English). (n.d.). https://dhsprogram.com/publications/publication-dhsg4-dhs-questionnaires-and-manuals.cfm

Earth Observation Group - Defense Meteorological Satellite Program, Boulder | ngdc.noaa.gov. (n.d.). https://ngdc.noaa.gov/eog/dmsp/downloadV4composites.html

Gallery. (2022, March 8). Cities at Night. https://citiesatnight.org/gallery/

G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition*

(CVPR), 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243. https://ieeexplore.ieee.org/document/8099726

Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (3rd ed.). O'Reilly Media. Random Forest

Hall, C. (2022, February 1). *Nighttime Lights*. Earthdata. https://www.earthdata.nasa.gov/learn/backgrounders/nighttime-lights

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).https://arxiv.org/pdf/1512.03385.pdf

Hong, Z., & Park, I. K. (2021). Comparative Analysis of Energy Poverty Prediction Models Using Machine Learning Algorithms'. *Journal of Korea Planning Association Vol*, *56*(5).

Hu, S., Ge, Y., Liu, M., Ren, Z., & Zhang, X. (2022). Village-level poverty identification using machine learning, high-resolution images, and geospatial data. *International Journal of Applied Earth Observation and Geoinformation*, *107*, 102694.

Li, Q., Yu, S., Échevin, D., & Fan, M. (2022). Is poverty predictable with machine learning? A study of DHS data from Kyrgyzstan. *Socio-Economic Planning Sciences*, *81*, 101195.

Madhur, L., & Praneeth, S. (2021, April 26). *RMSE: What does it mean?* Medium. Retrieved December 15, 2021, from https://medium.com/@mygreatlearning/rmse-what-does-it mean-2d446c0b1d0e

Manas, E., & Chandra, R. (2018, February 22). *Mean absolute error ~ mae [machine learning(ml)]*. Medium. Retrieved December 15, 2021, from https://medium.com/@ewuramaminka/mean-absolute-error-mae-machine-learning-ml b9b4afc63077

Ncei. (n.d.). *Version 4 DMSP-OLS nighttime lights time series*. Earth Observation Group - Defense Meteorological Satellite Progam, Boulder. Retrieved October 16, 2022, from https://ngdc.noaa.gov/eog/dmsp/downloadV4composites.html

Overview | Maps Static API |. (n.d.). Google Developers. https://developers.google.com/maps/documentation/maps-static/overview

Roser, M. (2021, November 22). *Extreme poverty: How far have we come, how far do we still have to go?* Our World in Data. Retrieved October 16, 2022, from https://ourworldindata.org/extreme-poverty-in-brief

S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

Survey, U.-. U. G. (n.d.). *EarthExplorer*. None. https://earthexplorer.usgs.gov/

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

The DHS Program - Available Datasets. (n.d.). https://dhsprogram.com/data/available-datasets.cfm

Tian, F., Wu, B., Zeng, H., Watmough, G. R., Zhang, M., & Li, Y. (2022). Detecting the linkage between arable land use and poverty using machine learning methods at global perspective. *Geography and Sustainability*, *3*(1), 7-20.

Verme, P. (2020). *Which Model for Poverty Predictions?* (No. 468). GLO Discussion Paper.

Wijaya, D. R., Paramita, N. L. P. S. P., Uluwiyah, A., Rheza, M., Zahara, A., & Puspita, D. R. (2020). Estimating city-level poverty rate based on e-commerce data with machine learning. *Electronic Commerce Research*, 1-27.

Xie, M., Jean, N., Burke, M., Lobell, D., & Ermon, S. (2016, March). Transfer learning from deep features for remote sensing and poverty mapping. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Xu, J., Song, J., Li, B., Liu, D., & Cao, X. (2021). Combining night time lights in prediction of poverty incidence at the county level. *Applied Geography*, *135*, 102552.

Y. Ni, X. Li, Y. Ye, Y. Li, C. Li and D. Chu, "An Investigation on Deep Learning Approaches to Combining Nighttime and Daytime Satellite Imagery for Poverty Prediction," in *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 9, pp. 1545-1549, Sept. 2021, doi: 10.1109/LGRS.2020.3006019.

Zixi, H. (2021, March). Poverty Prediction Through Machine Learning. In *2021 2nd International Conference on E-Commerce and Internet Technology (ECIT)* (pp. 314-324). IEEE.

Appendix A

System Testing

System testing is about testing whether the system is working as expected. In this appendix we will present the flow of executing websites from the giving input to the prediction of the wealth index. Figures consist of the steps of the login page, uploading images and predicting the wealth index of uploaded images. Once an image has been uploaded to the portal, it will then go ahead and predict the wealth index, the image can be classified as “Dim”, “Medium” and “Bright”.

Figure A1

Figure A2

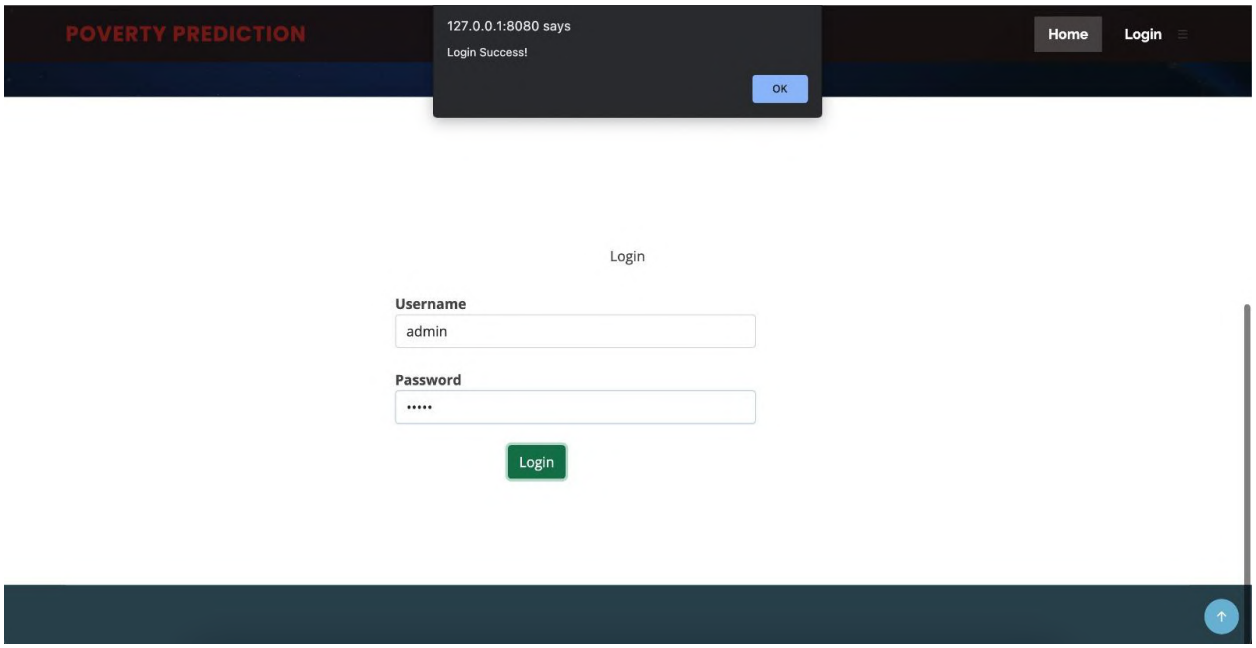


Figure A3

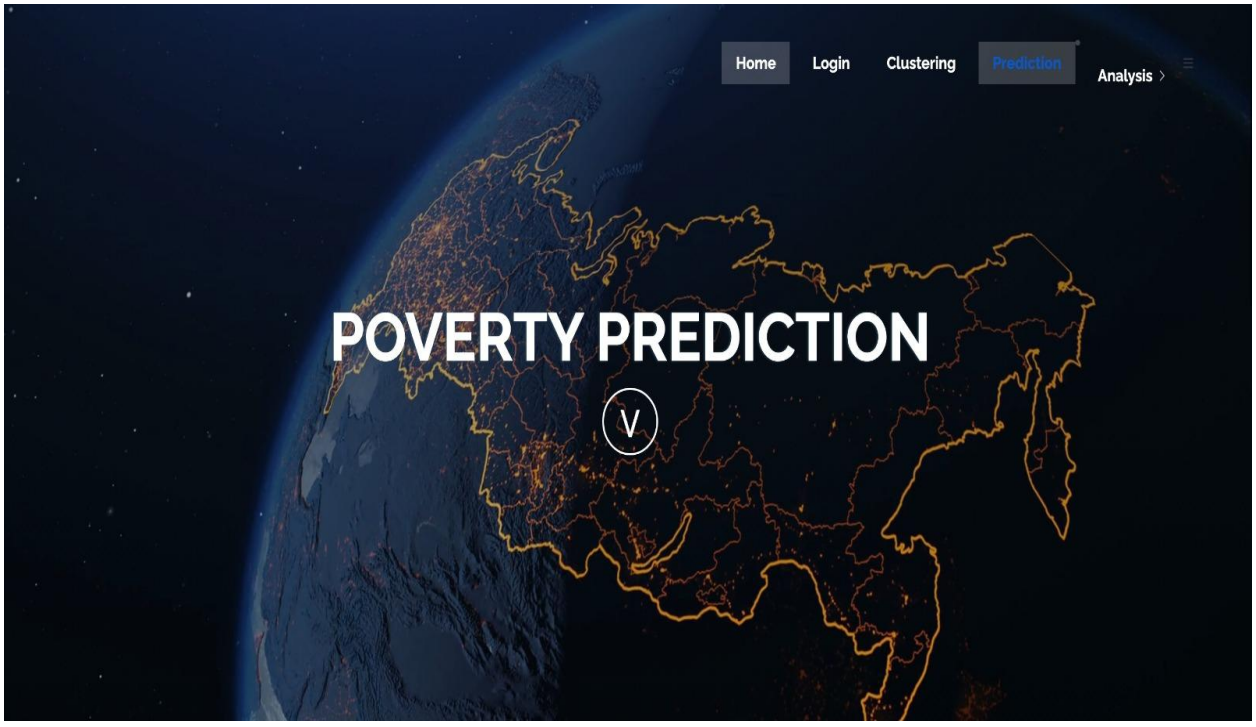


Figure A4

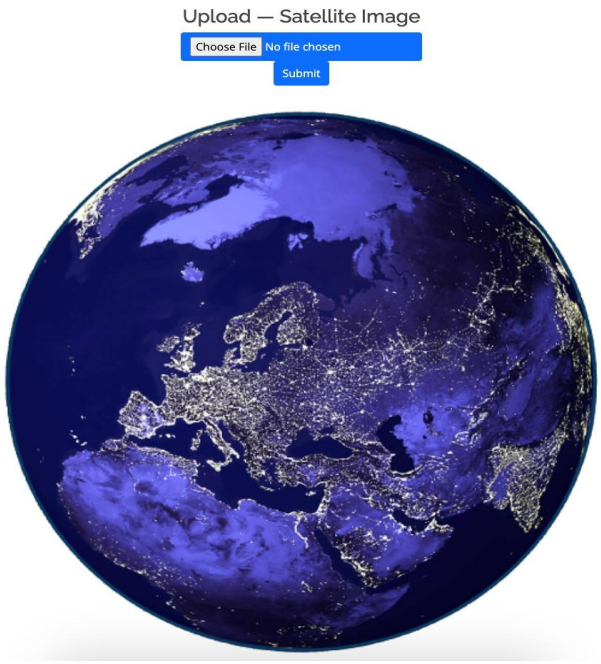


Figure A5

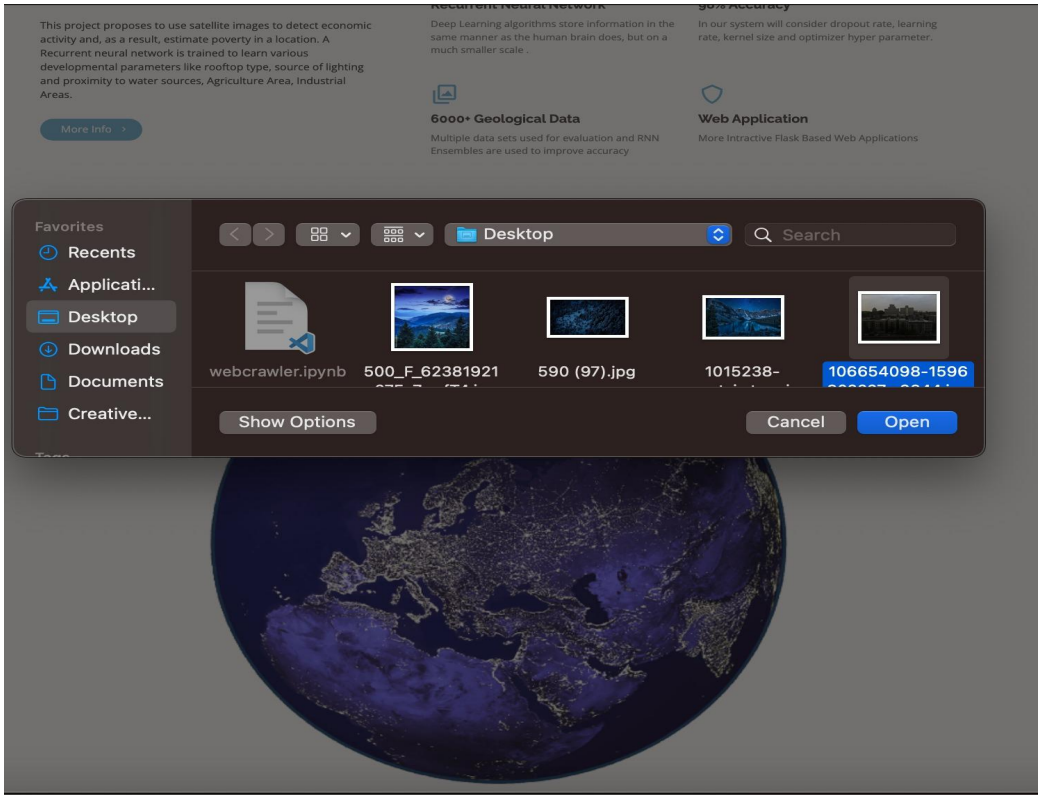


Figure A6

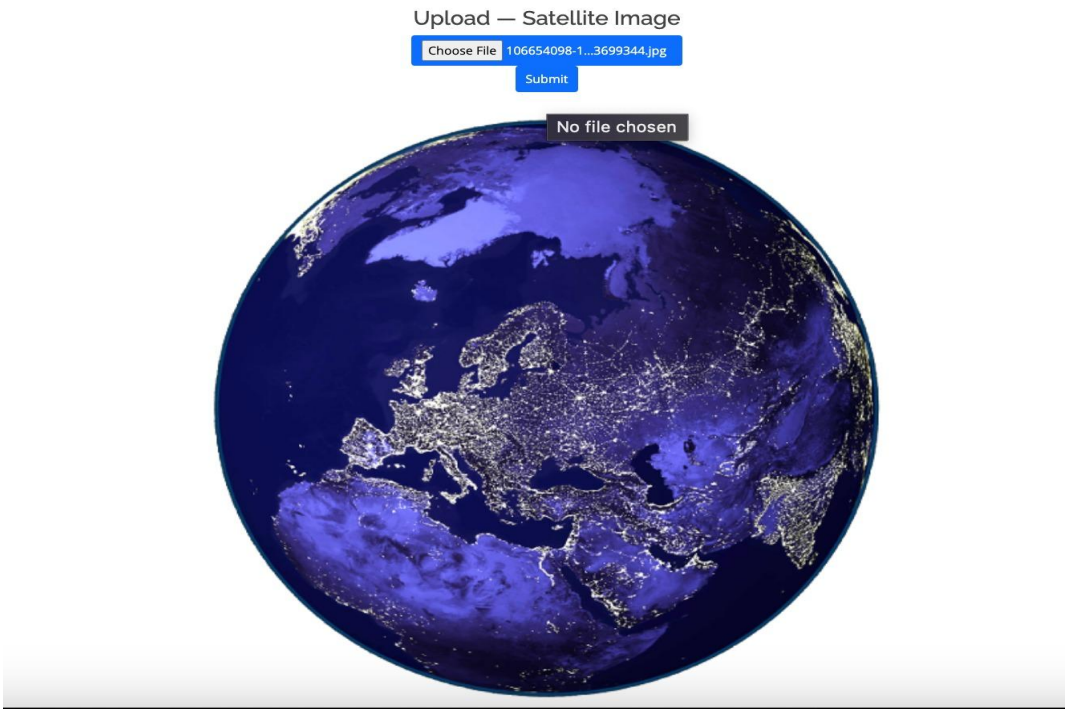


Figure A7

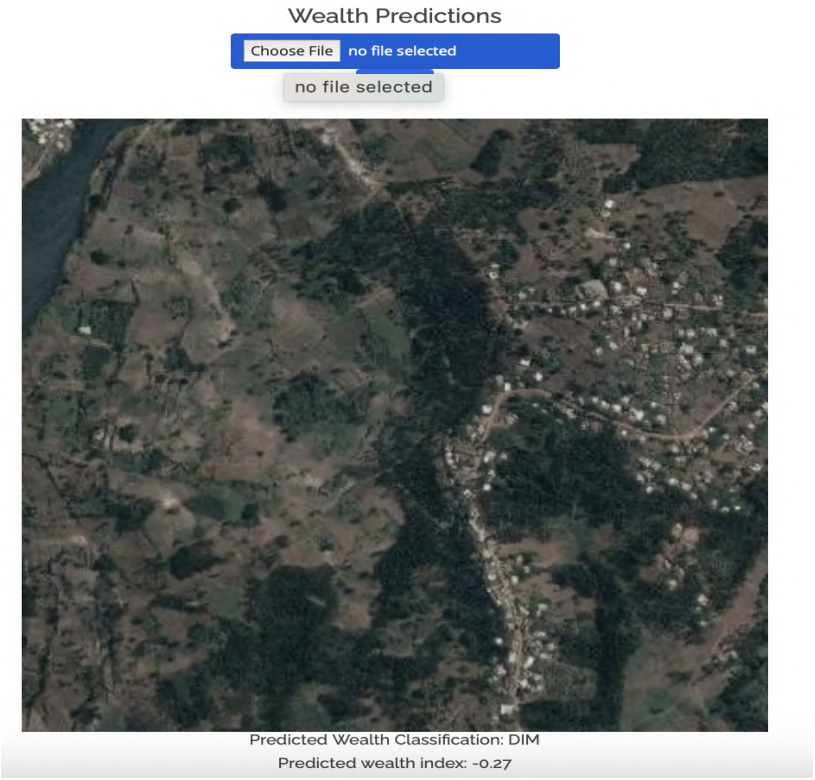
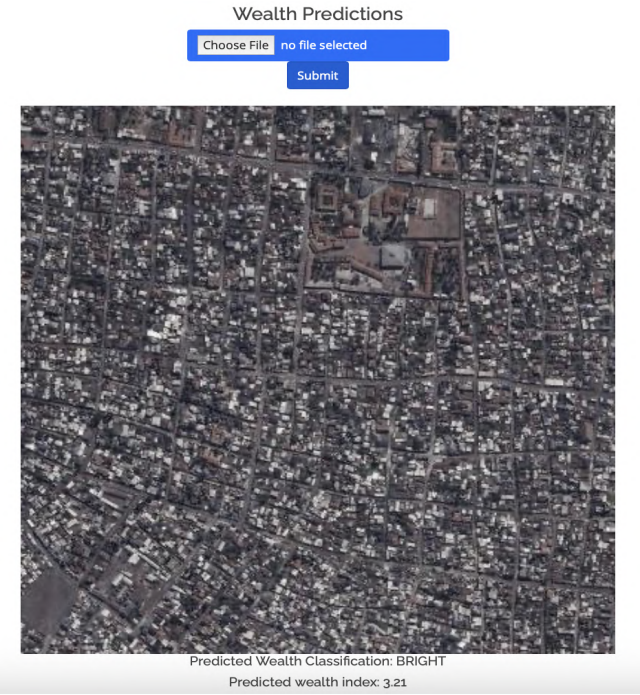


Figure A8

Figure A9



Appendix B

Project Data Source and Management Store

For the project we mainly used satellite images, which were downloaded using a python script from the Google maps static API by providing the shapes and boundary and the geographic coordinates which is latitude and longitude for the targeted images. Link to download the day time satellite images from google map is

https://developers.google.com/maps/documentation/maps-static

For the night time, we download a tgif file from the National Oceanic and atmospheric administration (NOAA). The file we used was a cloud free composite which was mapped by using DMSP-OLS resolution data for a particular calendar year. The file image is of 30 seconds arc second grids and the latitude spanning form -180 to 180 and latitude degrees from -65 to 75. To download the file the link is given below.

https://ngdc.noaa.gov/eog/dmsp/downloadV4composites.html

We also collected survey data for the locations which we were using to download the images. The survey data contained the geographic coordinates and other information such as living conditions, wealth index of the cluster which we used to verify our model and it acted as a ground truth for our project. The data which we downloaded was only available for legitimate research purposes, so we had to submit an abstract for access. The link to download the dataset is given below

https://dhsprogram.com/data/available-datasets.cfm

For storing all the data, we decided to use the google drive platform and created a shared drive which can be accessed by all the team members and it also helped us to write the code and store the intermediate files generated during the code execution.

Link to our dataset and project related files

https://drive.google.com/file/d/1rZWg5OyW3VKjWUNwggbbefTn7j7W0vjM/view?usp=share_link

Appendix C

Project Program Source Library, Presentation, and Demonstration

The below given links contain the source code for our UI, model, data processing, data collection and exploration. It also contains presentations we used throughout the project and demonstration videos of our project.

Project Program Source Code

https://drive.google.com/drive/folders/13LgaEeo_06gqSC_NjftBaR75KS_AZTGc?usp=share_link

Presentation and Demos

https://drive.google.com/drive/folders/1PmCzG35F9ZzYrGGbpCWRW8T4nOMLHAxI?usp=share_link

Workbooks

https://drive.google.com/drive/folders/16T8QGOYcM_xFcauzieM2lZkETcMzqtmp?usp=share_link

Reports

https://drive.google.com/drive/folders/1Wd32E83uc3PFdJVl2iB1y97v2UqtyXQ1?usp=share_link