**Project Report**

**Title : Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy**

**Prepared By:**

|  |  |  |
| --- | --- | --- |
| **Name** | **College** | **Mob.No** |
| Suguna.R | National Engineering College | 9994713996 |

**ABSTRACT**

Diabetic Retinopathy (DR) poses a substantial threat to the vision health of individuals with diabetes mellitus, emphasizing the critical need for early detection to mitigate its potentially irreversible consequences. This project explores the integration of deep learning techniques into fundus image analysis to develop a robust and efficient system for the early detection of DR. Leveraging the power of convolutional neural networks (CNNs) and advanced image processing algorithms, the proposed deep learning model aims to autonomously analyze retinal images, identify subtle signs of DR at its incipient stages, and contribute to timely intervention. The core objective is to address the inherent challenges of the current manual diagnostic processes, such as subjectivity, inter-observer variability, resource intensity, and limited scalability. The deep learning model presented herein seeks to provide a standardized and objective approach to DR detection, reducing dependence on individual expertise and facilitating consistent, accurate, and rapid assessments. The methodology involves training the deep learning model on a diverse dataset of retinal fundus images, encompassing various stages of DR. The model learns intricate patterns and features indicative of early DR manifestations, enabling it to generalize well to unseen data. Rigorous validation and testing procedures are implemented to ensure the model's reliability and generalizability across different patient demographics and imaging conditions. The anticipated impact of this project is twofold: firstly, the early detection of DR at its nascent stages, enabling timely intervention and management; and secondly, the establishment of an efficient and scalable diagnostic solution that transcends the limitations of manual assessments. The deep learning model, once validated and integrated into clinical workflows, has the potential to revolutionize DR diagnosis, significantly improving patient outcomes and alleviating the burden on healthcare resources. Through the convergence of medical expertise and technological innovation, this project endeavors to contribute to the advancement of ophthalmic care, demonstrating the transformative potential of deep learning in early DR detection. The outcomes of this research not only hold promise for enhancing diagnostic accuracy but also underscore the pivotal role of artificial intelligence in shaping the future of healthcare, particularly in the context of diabetic retinopathy.

**1. Introduction:**

Artificial Intelligence (AI) and Machine Learning (ML) are transformative technologies that have gained significant prominence in recent years. These technologies empower machines to learn from data, recognize patterns, and make intelligent decisions without explicit programming. Leveraging the capabilities of AI and ML in conjunction with cloud computing has become a powerful approach to address complex computational tasks and enhance scalability.

**1.1 Overview:**

**AI and ML:**

AI: Artificial Intelligence refers to the development of computer systems that can perform tasks that typically require human intelligence. This includes tasks such as problem-solving, understanding natural language, and recognizing patterns.

ML: Machine Learning is a subset of AI that focuses on the development of algorithms and statistical models that enable computers to improve their performance on a specific task through learning from data.

Cloud Computing: Cloud computing involves the delivery of computing services (including storage, processing power, and applications) over the internet, allowing users to access and use these resources on-demand without the need for physical infrastructure.

**Key Characteristics:**

On-Demand Self-Service

Broad Network Access

Resource Pooling

Rapid Elasticity

Measured Service

**AI/ML in the Cloud:**

**Data Storage and Processing:** Cloud platforms provide scalable and cost-effective storage solutions, enabling organizations to store and manage vast amounts of data. This is crucial for ML, as large datasets are often required for training models.

**Computation Power:** ML models, especially deep learning models, demand substantial computational resources. Cloud services offer powerful and scalable computing resources, allowing organizations to train complex models efficiently.

**Scalability:** Cloud environments enable easy scaling of AI/ML workloads based on demand. Organizations can scale up or down resources dynamically, optimizing costs and performance.

**Accessibility:** Cloud-based AI/ML services are accessible from anywhere with an internet connection, facilitating collaboration and remote work.

**Managed Services:** Cloud providers offer managed AI/ML services that abstract the complexities of infrastructure management. This allows organizations to focus on model development and deployment rather than infrastructure maintenance.

**CHAPTER 2**

**LITERATURE REVIEW**

This chapter discusses about the various methodologies used to perform the image analysis of early detection of diabetic retinopathy.

* 1. **Residual Neural Networks (ResNet)**

ResNet is a type of CNN that introduced residual learning, addressing the challenges of training very deep networks. ResNet architectures have been applied to fundus image analysis to capture intricate features associated with diabetic retinopathy. The core innovation in ResNet is the introduction of residual blocks. A residual block consists of a shortcut connection (also known as a skip or identity connection) that bypasses one or more layers. Instead of learning the direct mapping, ResNet learns the residual mapping— the difference between the input and output. The output of a residual block is the sum of the input and the learned residual.

ResNet encourages the stacking of numerous residual blocks to create very deep neural networks. Traditional deep networks faced challenges like vanishing or exploding gradients, making training difficult. Residual connections mitigate these issues by enabling the gradient to flow directly through the shortcut connections, ensuring smoother optimization. The use of identity mapping in residual connections allows the network to learn an identity function when needed. If a particular set of layers in a residual block doesn't contribute to improving the performance of the network, the weights associated with those layers can be adjusted to approximate an identity mapping. This flexibility contributes to the efficient training of deep networks.

ResNet employs global average pooling as the final layer, replacing traditional fully connected layers. This helps reduce the number of parameters in the network and ensures that the model is more robust to spatial translations. For very deep networks (e.g., ResNet-50, ResNet-101, ResNet-152), a bottleneck architecture is introduced in residual blocks to reduce computational complexity. It involves using 1x1 convolutions to decrease and then increase the dimensions of the input, reducing the number of computations while maintaining expressive power.

* 1. **Inception Networks (GoogLeNet)**

Inception networks, known for their inception modules, have been utilized in DR detection. These architectures are designed to capture multi-scale features, making them effective in analyzing retinal images with varying lesion sizes. The inception module is the central architectural component of GoogLeNet. Instead of relying on a single receptive field size, inception modules use multiple filter sizes (1x1, 3x3, 5x5) and pooling operations in parallel. This allows the network to capture both fine-grained and coarser features simultaneously, enabling it to learn a diverse set of features at different scales. Inception modules extensively use 1x1 convolutions to reduce the dimensionality of input feature maps before applying larger convolutions. These 1x1 convolutions serve two primary purposes: dimension reduction and introducing non-linearity through activation functions. They also help control the computational cost of the network.

The inception module performs parallel convolutions with different filter sizes and pooling operations. This parallel structure allows the network to learn features at various spatial scales, capturing both fine details and global context in an image. Inception Networks use max-pooling layers for spatial reduction. This helps decrease the spatial dimensions of the feature maps, reducing the computational burden while maintaining important information.

To address the vanishing gradient problem during training of deep networks, GoogLeNet introduces auxiliary classifiers. These classifiers are placed at intermediate layers and provide additional supervision during training. The auxiliary classifiers contribute to mitigating the risk of gradient vanishing and facilitate more stable training. Similar to ResNet, GoogLeNet uses global average pooling as its final layer. This layer reduces the spatial dimensions of the feature maps to a single value for each feature channel, serving as a form of regularization and reducing the number of parameters in fully connected layers

* 1. **Capsule Networks (CapsNets)**

CapsNets, an alternative to traditional CNNs, focus on capturing hierarchical relationships between features. They have shown promise in tasks related to understanding spatial hierarchies in medical images, including fundus images for DR detection. The fundamental building blocks in CapsNets are capsules. Unlike neurons in traditional networks, capsules are designed to represent entities in the input data and encode various attributes, such as orientation, scale, and position. Capsules work collaboratively to encapsulate the instantiation parameters of specific entities, providing a more nuanced representation compared to traditional neural network units.

CapsNets introduce dynamic routing mechanisms, specifically dynamic routing-by-agreement. In this process, capsules iteratively adjust their connection weights based on the agreement between the current and target activations. Dynamic routing enables capsules to reach a consensus on the instantiation parameters of the entities they represent, allowing for better representation of hierarchical relationships. Capsules output pose parameters that capture the instantiation details of entities. These parameters include information about the orientation, scale, and position of the represented entity. CapsNets leverage this pose information to understand spatial hierarchies and variations in the appearance of features. To ensure that the outputs of capsules are within a valid range, a squashing function is applied. This function compresses the raw output vectors of capsules, maintaining their relative proportions while ensuring the output lies between 0 and 1.

At the lower levels of the network, primary capsules capture basic features from the input data. They serve as the initial building blocks, extracting low-level information such as edges and textures. CapsNets employ a dynamic routing algorithm to determine the flow of information between capsules in different layers. During training, capsules in higher layers adjust their connection weights based on the agreement with lower-layer capsules. This iterative process enables capsules to reach a consensus on the instantiation parameters of the entities they represent.

* 1. **Generative Adversarial Networks (GANs):**

GANs have been explored for generating synthetic retinal images or augmenting existing datasets. They can contribute to training deep learning models by providing diverse examples of retinal abnormalities. The generator is a neural network responsible for generating new data samples. It takes random noise as input and transforms it into data that ideally should be indistinguishable from real data. The generator's goal is to create data that is realistic enough to deceive the discriminator.

The discriminator is another neural network that evaluates the authenticity of a given data sample. Its role is to distinguish between real data and data generated by the generator. The discriminator is trained to correctly classify samples as either real or fake. The training process involves a continuous interplay between the generator and the discriminator. The generator tries to improve its ability to produce realistic data, while the discriminator seeks to become more adept at distinguishing real from fake. This adversarial training creates a feedback loop, leading to improvements in both networks.

The objective of GANs is framed as a minimax game. The generator aims to minimize the likelihood of the discriminator correctly classifying generated samples as fake, while the discriminator seeks to maximize its accuracy in distinguishing real from generated samples. The generator and discriminator use different loss functions. The generator's loss is based on the likelihood that the generated samples are classified as real by the discriminator. The discriminator's loss is a combination of the errors in classifying real and fake samples.

During training, the generator and discriminator are updated iteratively. The generator creates new samples, and the discriminator evaluates them. The gradients from the discriminator's evaluation are then used to update the generator, improving its ability to generate more convincing samples. GANs are susceptible to a phenomenon known as mode collapse, where the generator produces a limited variety of samples, ignoring some modes in the data distribution. Researchers have introduced techniques like minibatch discrimination and spectral normalization to address this issue. GANs have seen various improvements and extensions, including Conditional GANs (cGANs) that generate samples conditioned on specific input information. GANs have applications in image generation, style transfer, super-resolution, image-to-image translation, and even text-to-image synthesis.

**CHAPTER 3**

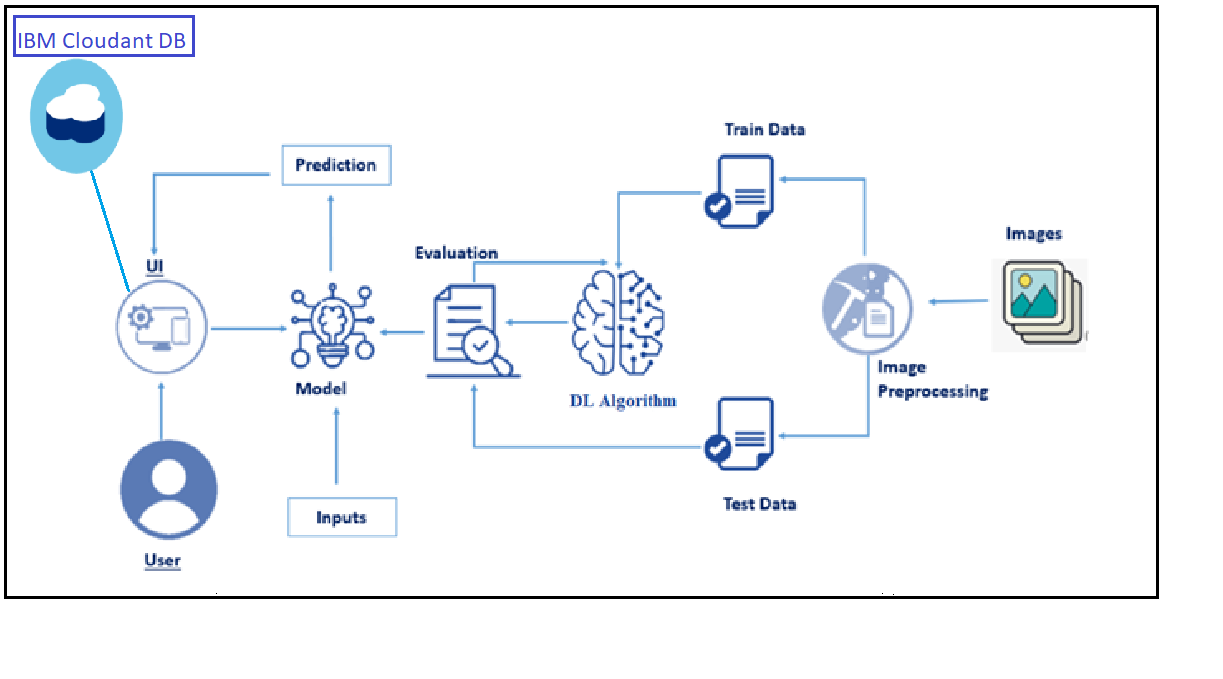
**SYSTEM DESIGN**

This chapter focuses on design of proposed methodology for the detection of diabetic retinopathy using deep learning.

**3.1 Proposed System**

The flow of the proposed system was shown in Figure 3.1. The steps consists of

* Data Collection
* Image Preprocessing
* Training and Testing
* Model building
* Performance analysis
* Prediction using GUI



**3.1 Hardware / Software designing**

To accomplish this, we have to complete all the activities and tasks listed below

* Data Collection.
  + Create a Train and Test path.
* Data Pre-processing.
* Import the required library
* Configure ImageDataGenerator class
* ApplyImageDataGenerator functionality to Trainset and Testset
* Model Building
  + Pre-trained CNN model as a Feature Extractor
  + Adding Dense Layer
  + Configure the Learning Process
  + Train the model
  + Save the Model
  + Test the model
* Cloudant DB
  + Register & Login to IBM Cloud
  + Create Service Instance
  + Creating Service Credentials
  + Launch Cloudant DB
  + Create Database
* Application Building
  + Create an HTML file
  + Build Python Code

**CHAPTER 4**

**IMPLEMENTATION**

This chapter focuses on the software requirements and python libraries needed to implement the proposed methodology.

**4.1 Strucure of implementation**

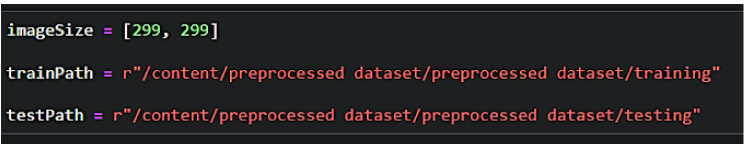
The proposed methodlogy implemented based on the following flow,

* Download the dataset
* Model Building
* Cloudant DB
* Application Building

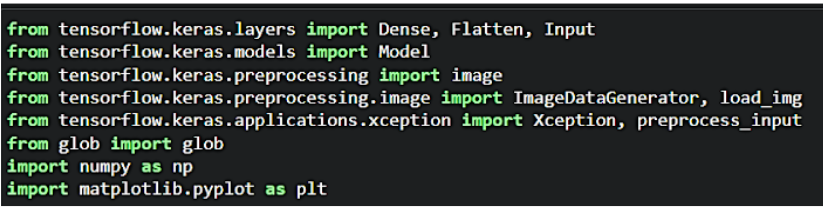
Dataset downloaded from Kaggle, and images are preprocessed, testing and training images are separated. Model built using Xception and Softmax. CloudantDB created then Python Flask application used to build web application.

**4.2 Dataset**

To build a DL model we have to split training and testing data into two separate folders. But In the project dataset folder training and testing folders are presented. So, in this case we just have to assign a variable and pass the folder path to it.Four different transfer learning models are used in our project and the best model (Xception) is selected.The image input size of xception model is 299, 299.



Import the necessary libraries as shown in the image.

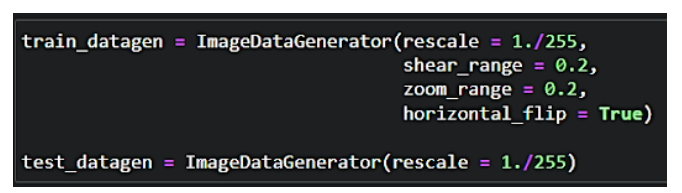


ImageDataGenerator class is instantiated and the configuration for the types of data augmentation

There are five main types of data augmentation techniques for image data; specifically:

* Image shifts via the width\_shift\_range and height\_shift\_range arguments.
* The image flips via the horizontal\_flip and vertical\_flip arguments.
* Image rotations via the rotation\_range argument
* Image brightness via the brightness\_range argument.
* Image zoom via the zoom\_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

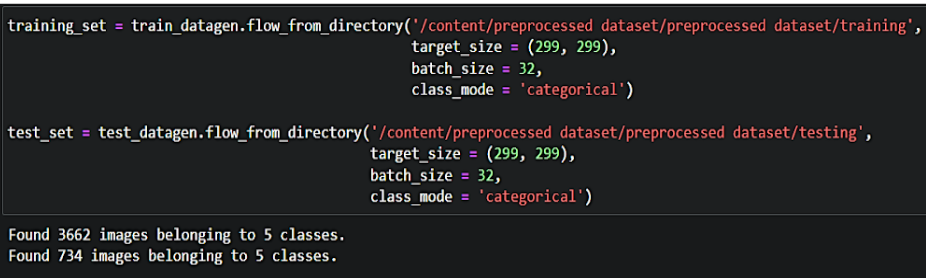


Let us apply ImageDataGenerator functionality to the Train set and Test set by using the following code. For Training set using flow\_from\_directory function.

This function will return batches of images from the subdirectories

Arguments:

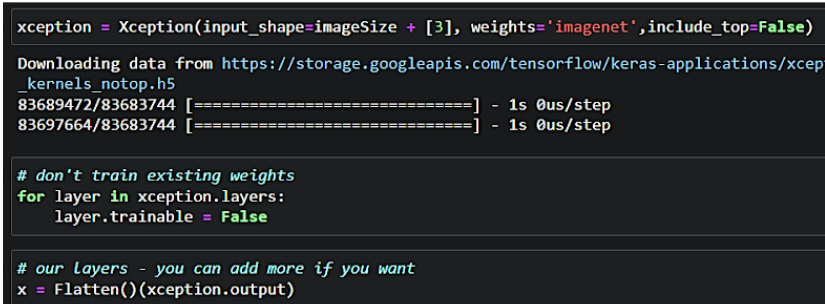
* directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
* batch\_size: Size of the batches of data which is  64.
* target\_size: Size to resize images after they are read from disk.
* class\_mode:



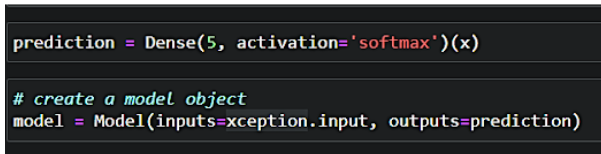
**4.3 Model Building**

For one of the models, we will use it as a simple feature extractor by freezing all the five convolution blocks to make sure their weights don’t get updated after each epoch as we train our own model.Here, we have considered images of dimension (229,229,3). Also, we have assigned include\_top = False because we are using convolution layer for features extraction and wants to train fully connected layer for our images classification(since it is not the part of Imagenet dataset)

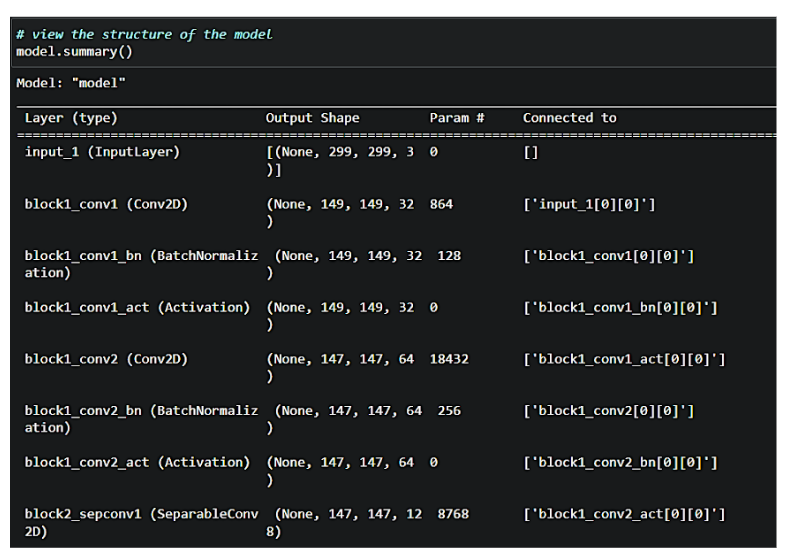
Flatten layer flattens the input. Does not affect the batch size



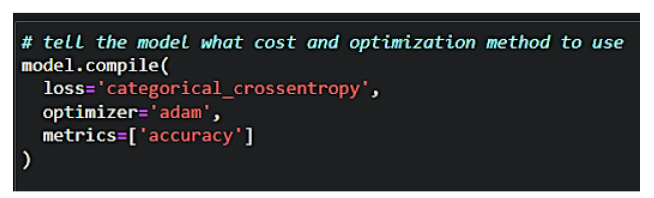
A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.Let us create a model object named model with inputs as xception.input and output as dense layer**.**



The number of neurons in the Dense layer is the same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities.  Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.



The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizerMetrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process.

 Now, let us train our model with our image dataset. The model is trained for 30 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 10 epochs and probably there is further scope to improve the model.

**fit\_generator** functions used to train a deep learning neural network

**Arguments:**

* steps\_per\_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of     steps\_per\_epoch as the total number of samples in your dataset divided by the batch size.
* Epochs: an integer and number of epochs we want to train our model for.
* validation\_data can be either:

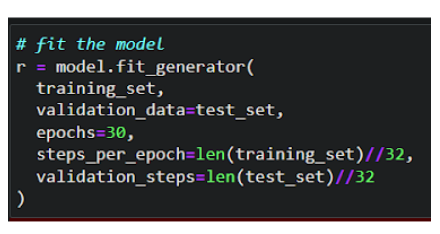
                      - an inputs and targets list

                      - a generator

                      - an inputs, targets, and sample\_weights list which can be used to evaluate

                        the loss and metrics for any model after any epoch has ended.

* validation\_steps: only if the validation\_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.



The model is saved with .h5 extension as follows An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.



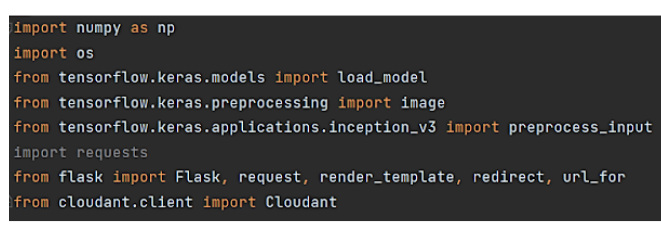
**4.3 Cloudant DB**

* Log in to IBM Cloud account, and click on Catalog
* To create the connection information that your application needs to connect to the instance, click New credential.
* Enter a name for the new credential in the Add new credential window.
* Accept the Manager role.
* (Optional) Create a service ID or have one automatically generated for you.
* (Optional) Add inline configuration parameters. This parameter isn't used by IBM Cloudant service credentials, so ignore it.
* Click Add.
* To see the credentials that are required to access the service, click the chevron.
* In order to manage a connection from a local system you must first initialize the connection by constructing a Cloudant client.We need to import the cloudant library.
* IBM Cloud Identity & Access Management enables you to securely authenticate users and control access to all cloud resources consistently in the IBM Bluemix Cloud Platform.

**4.4 Application Building**

In python flask HTML files, CSS and required images are placed in temlpate and static folder. The main application is created by the following steps,

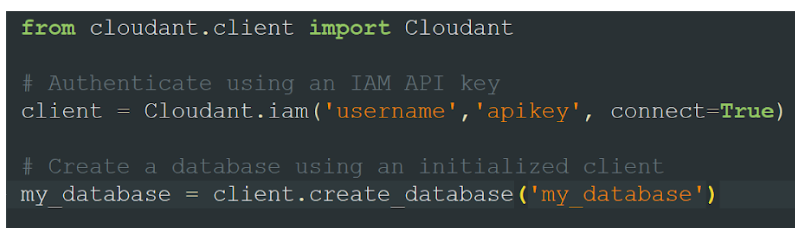
* Import Libraries
* Create Database
* Render HTML pages
* Configure the separate pages
* Predcition on UI



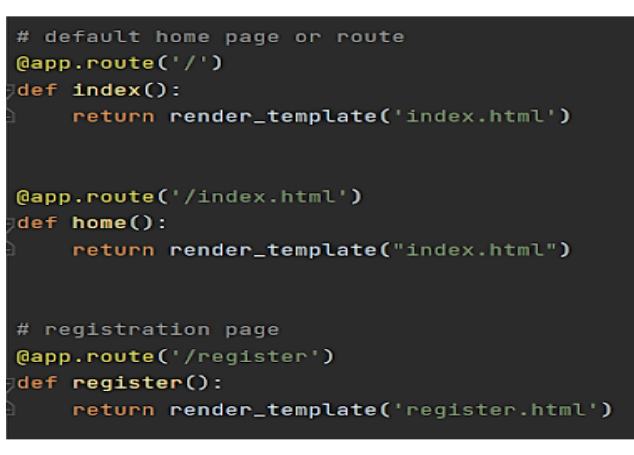
Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (\_\_name\_\_) as argument.



Create a database using an initiated client.



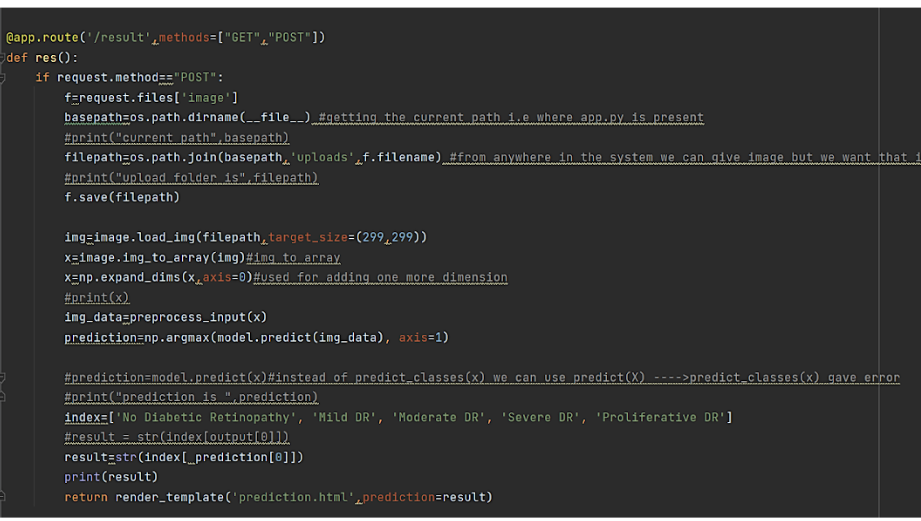
Render HTML page



Based on user input into the registration form we stored it on data dictionary then we can validate the data using \_id parameter with user input that we can store it on query variable then we can validate by passing the query variable into the my\_database.get\_user\_result() method. Then we can check the docs length by using len(docs.all()) function. If the length of docs is 0 then user will register successfully on the platform and user data will store on the database. Otherwise its shows the message as user already registered please login and use our web application for DR prediction. Based on user input into the login form we stored user id and password into the (user,passw) variables. Then we can validate the credentials using \_id parameter with user input that we can store it on query variable then we can validate by passing the query variable into the my\_database.get\_user\_result() method. Then we can check the docs length by using len(docs.all()) function. If the length of doc is 0 then it means username is not found. Otherwise its validate the  data that is stored on the database and check the username & password. If it's matched then the user will be able to login and use our web application for DR prediction. Otherwise the user needs to provide correct credentials.



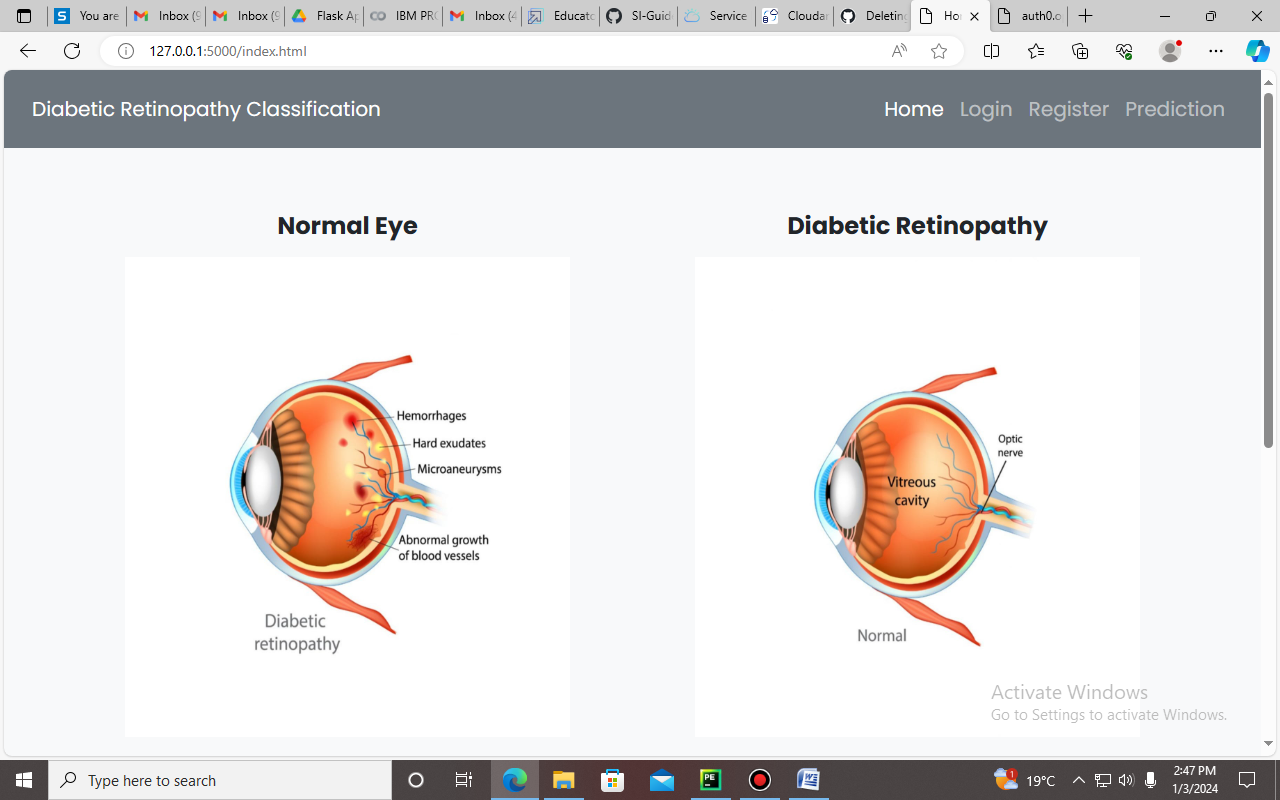
The image is selected from uploads folder. Image is loaded and resized with load\_img() method. To convert image to an array, img\_to\_array() method is used and dimensions are increased with expand\_dims() method. Input is processed for xception model and predict() method is used to predict the probability of classes. To find the max probability np.argmax is used.

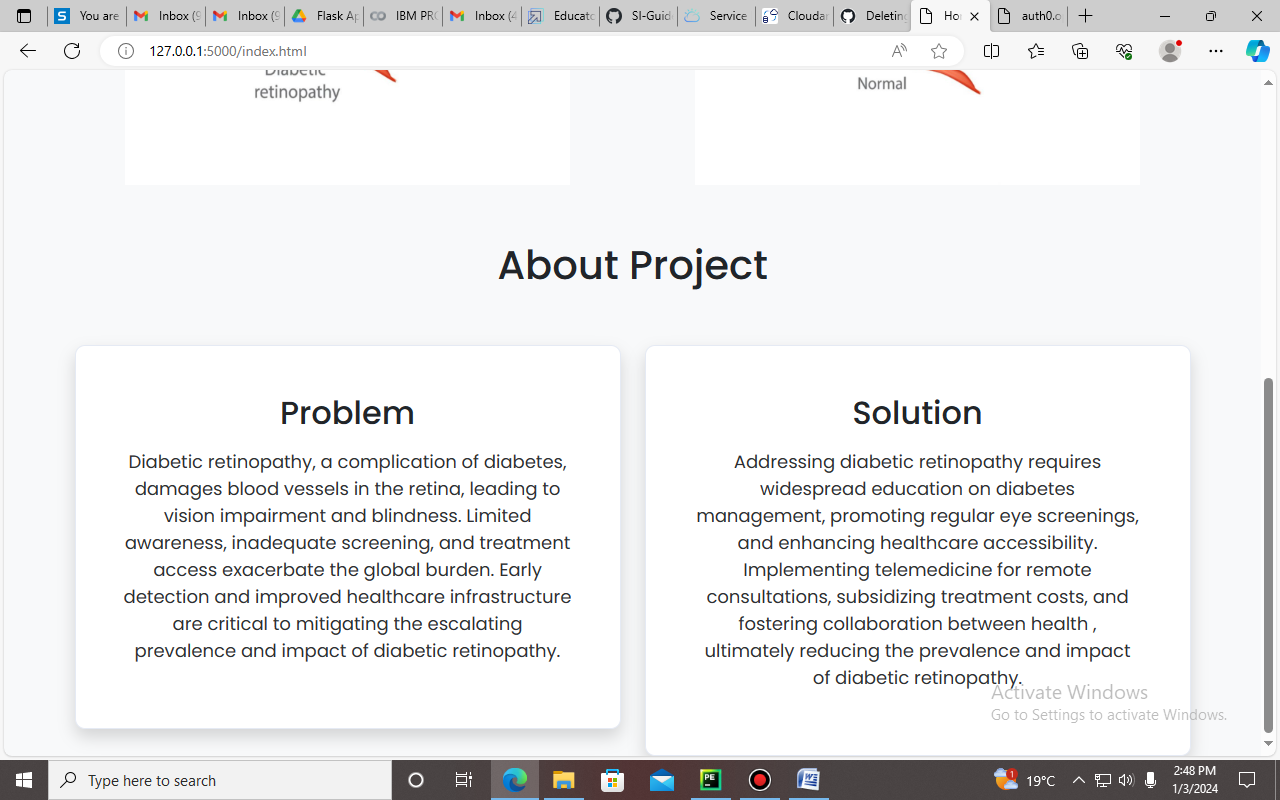


**CHAPTER 5**

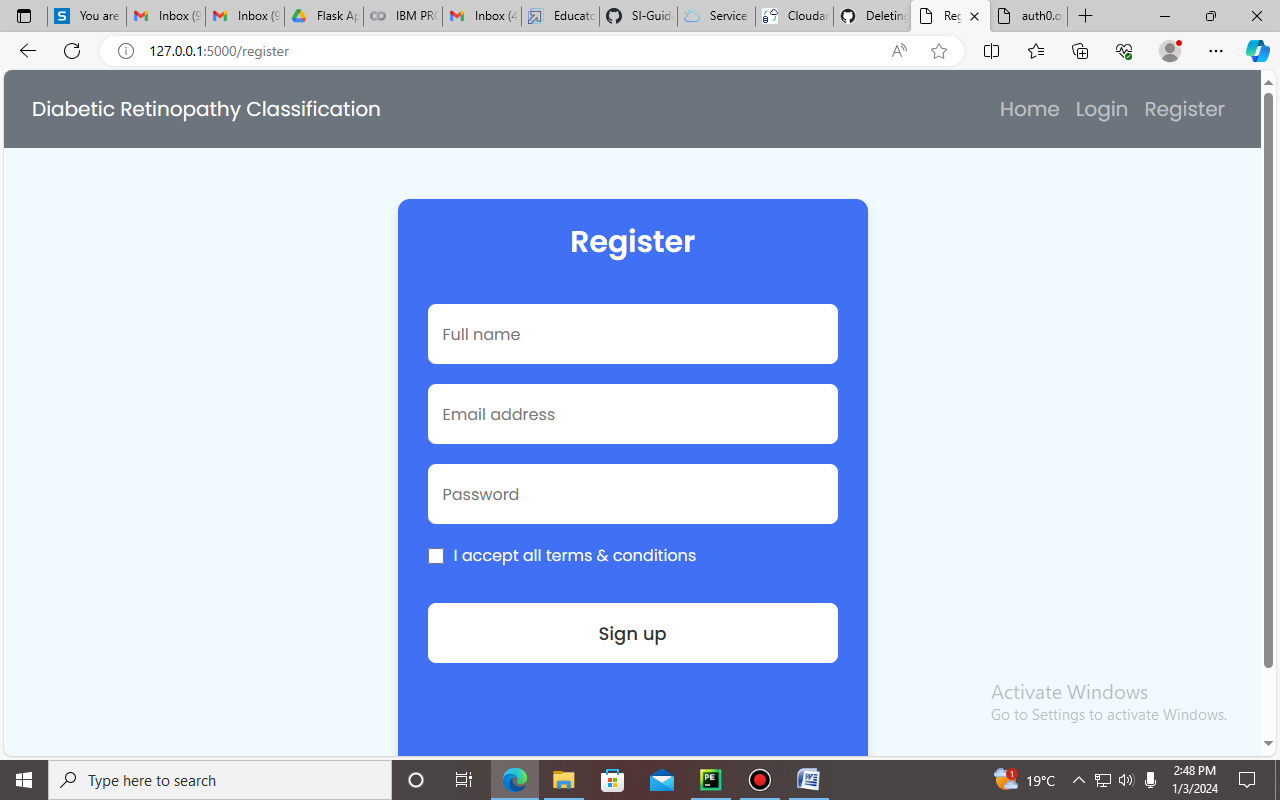
**RESULTS**

1. FLASK Application Index.html

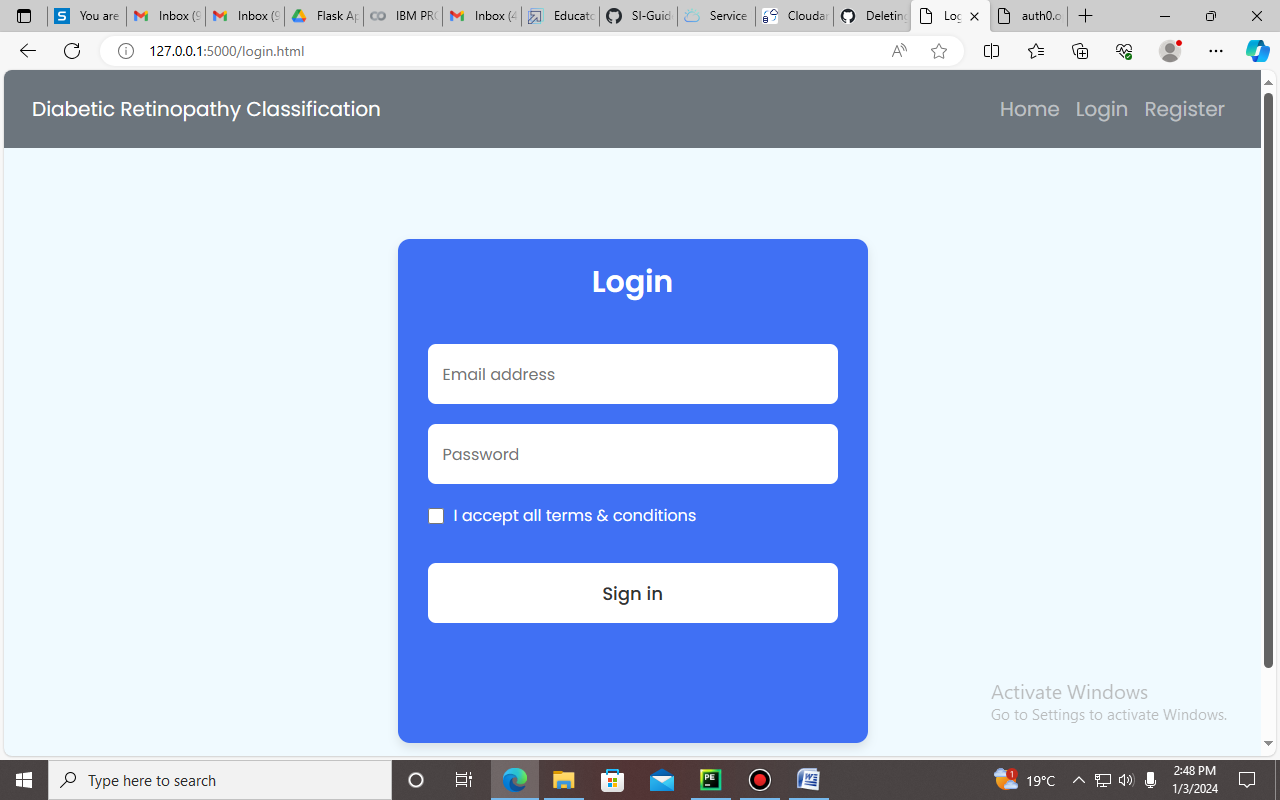




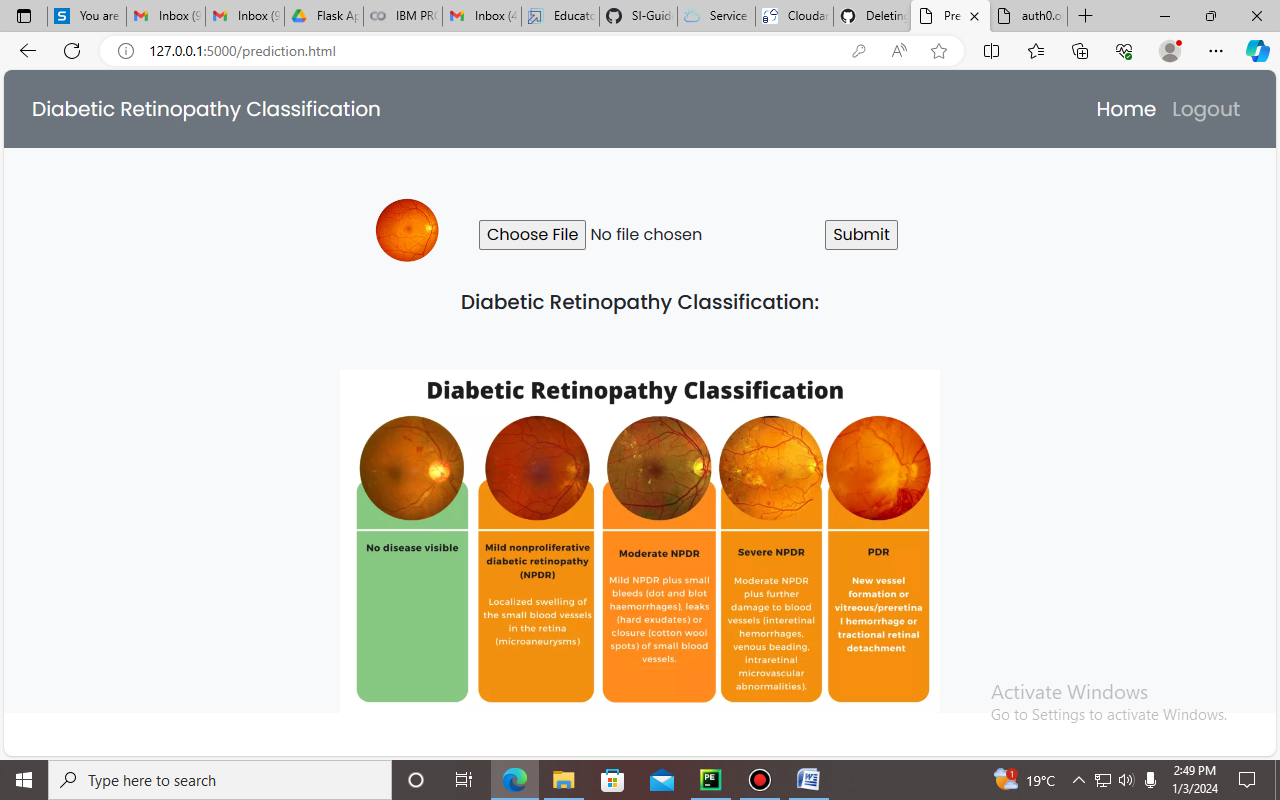
1. Register.html



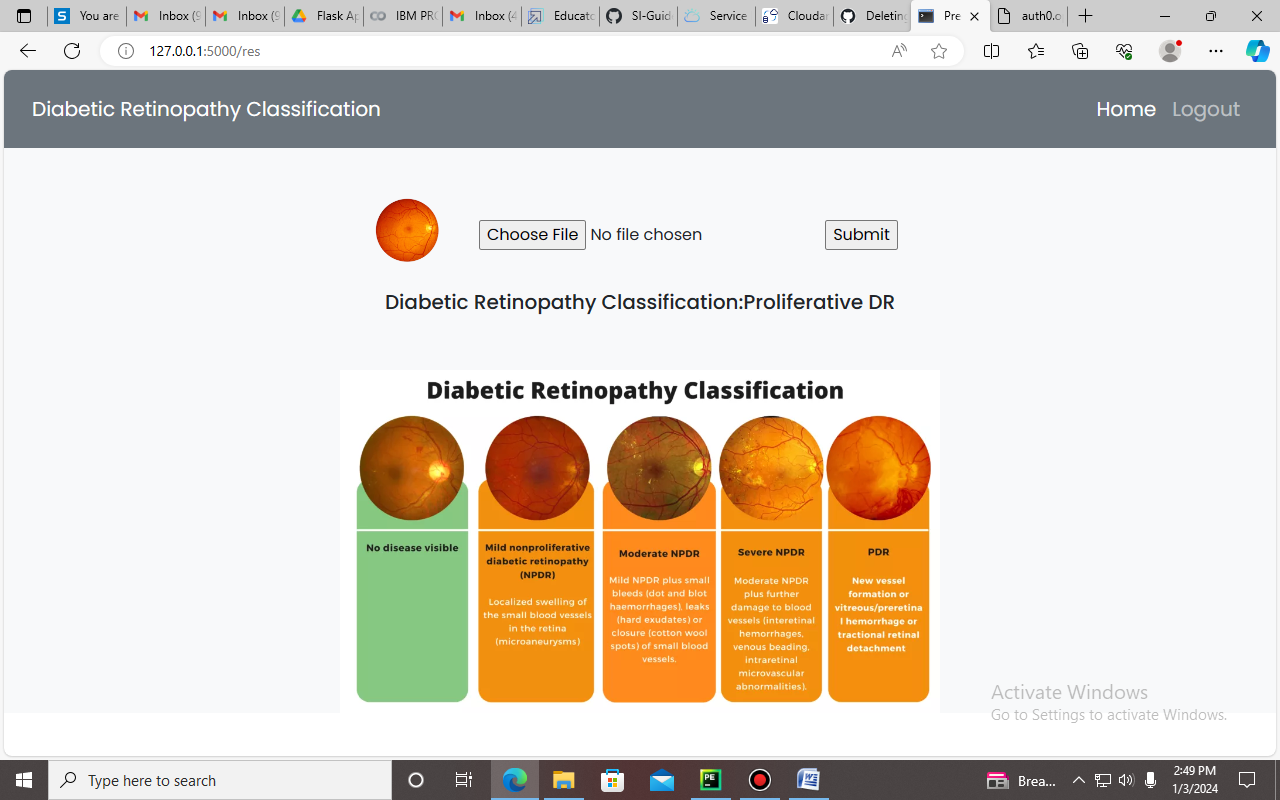
1. Login.html

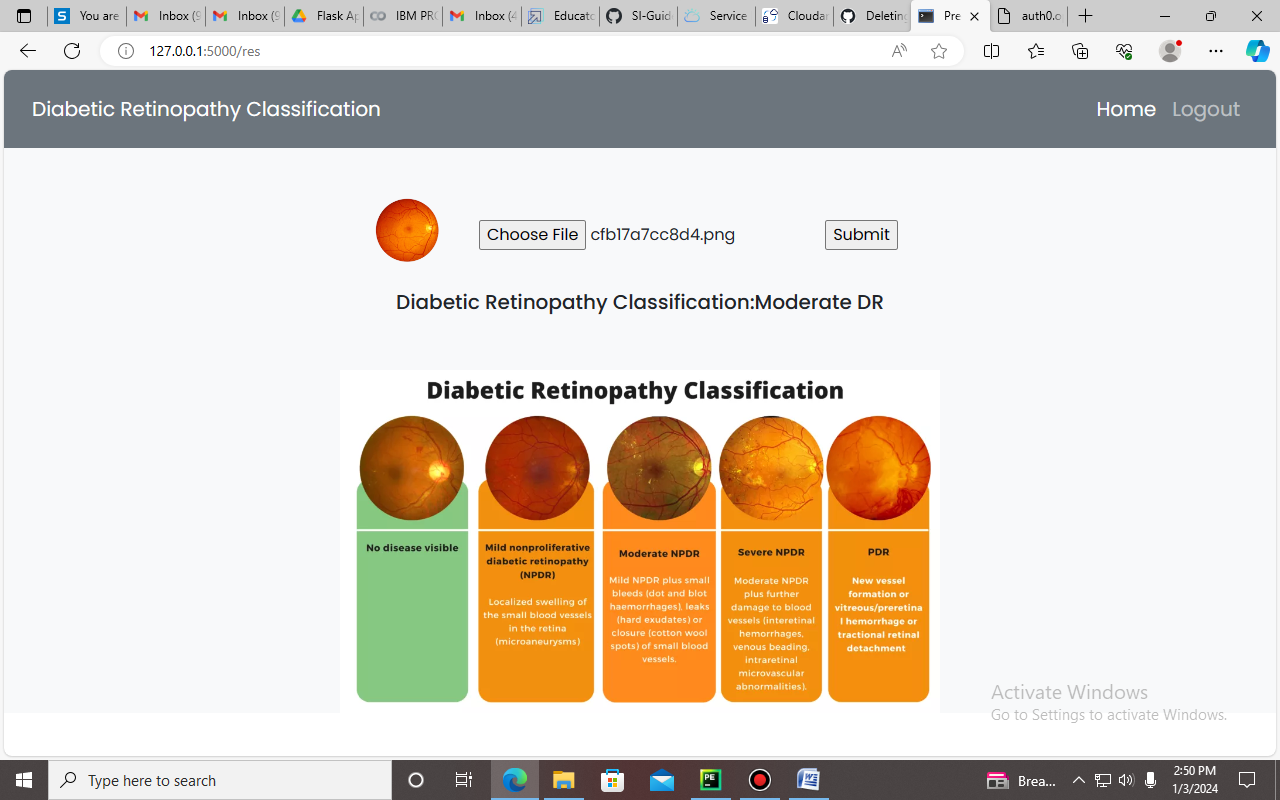


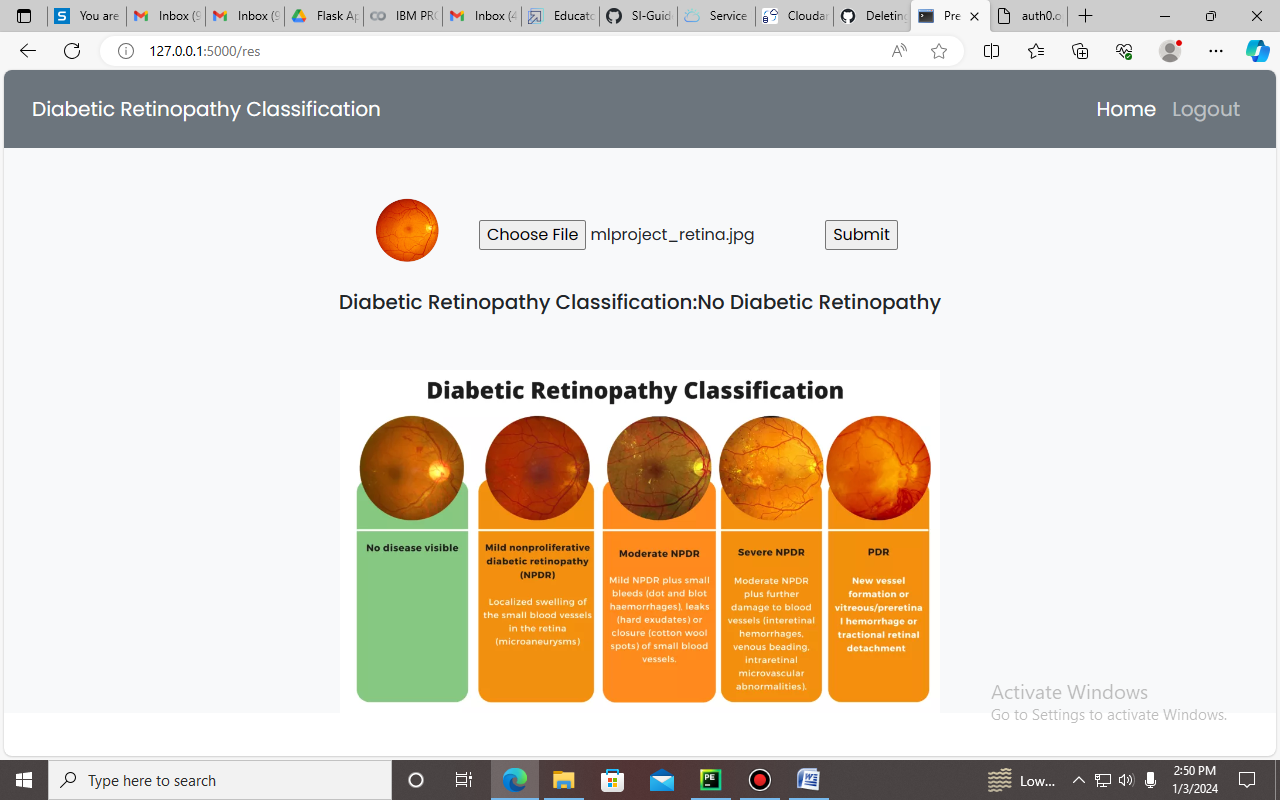
1. Classification.html



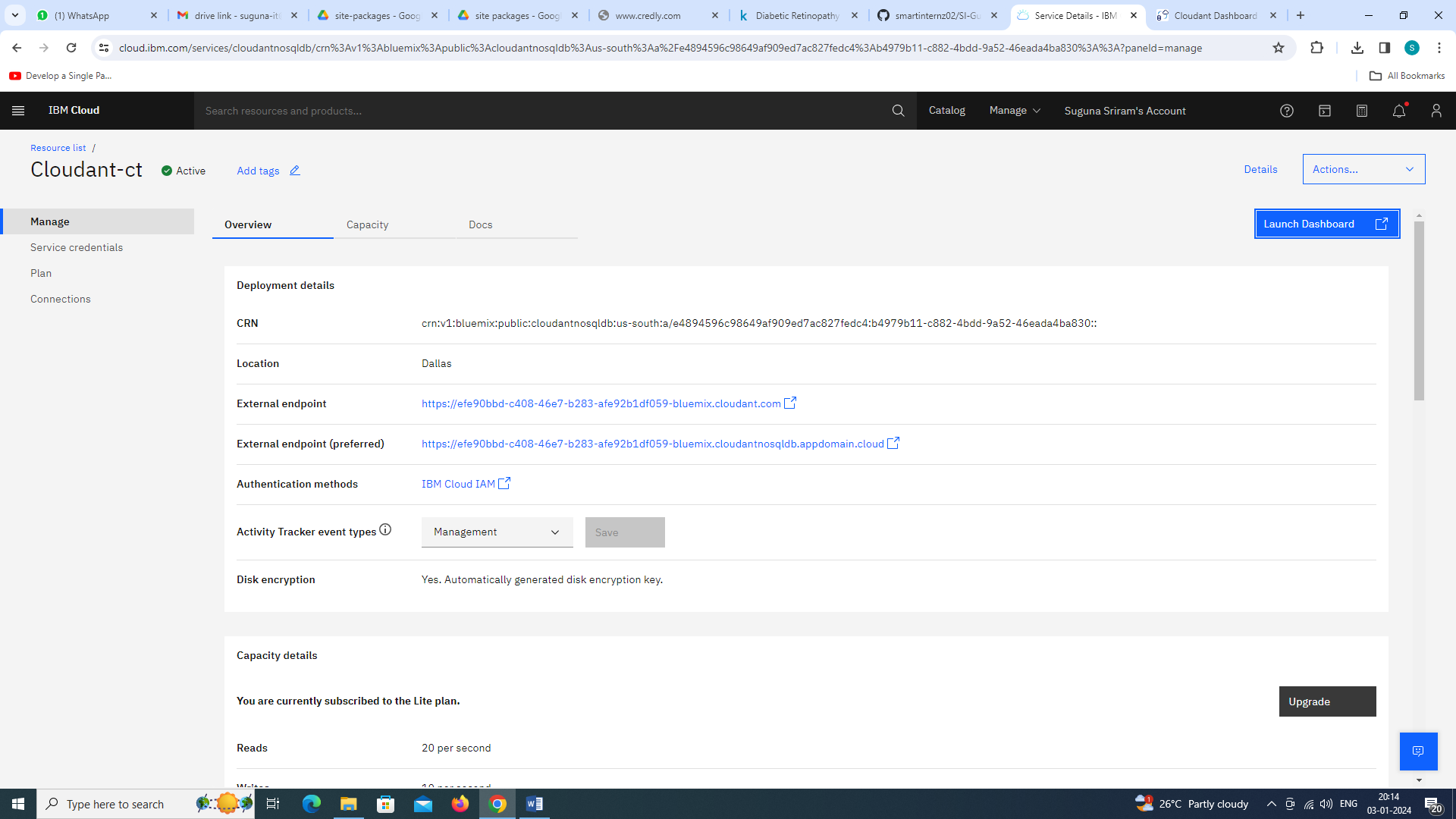
1. Prediction

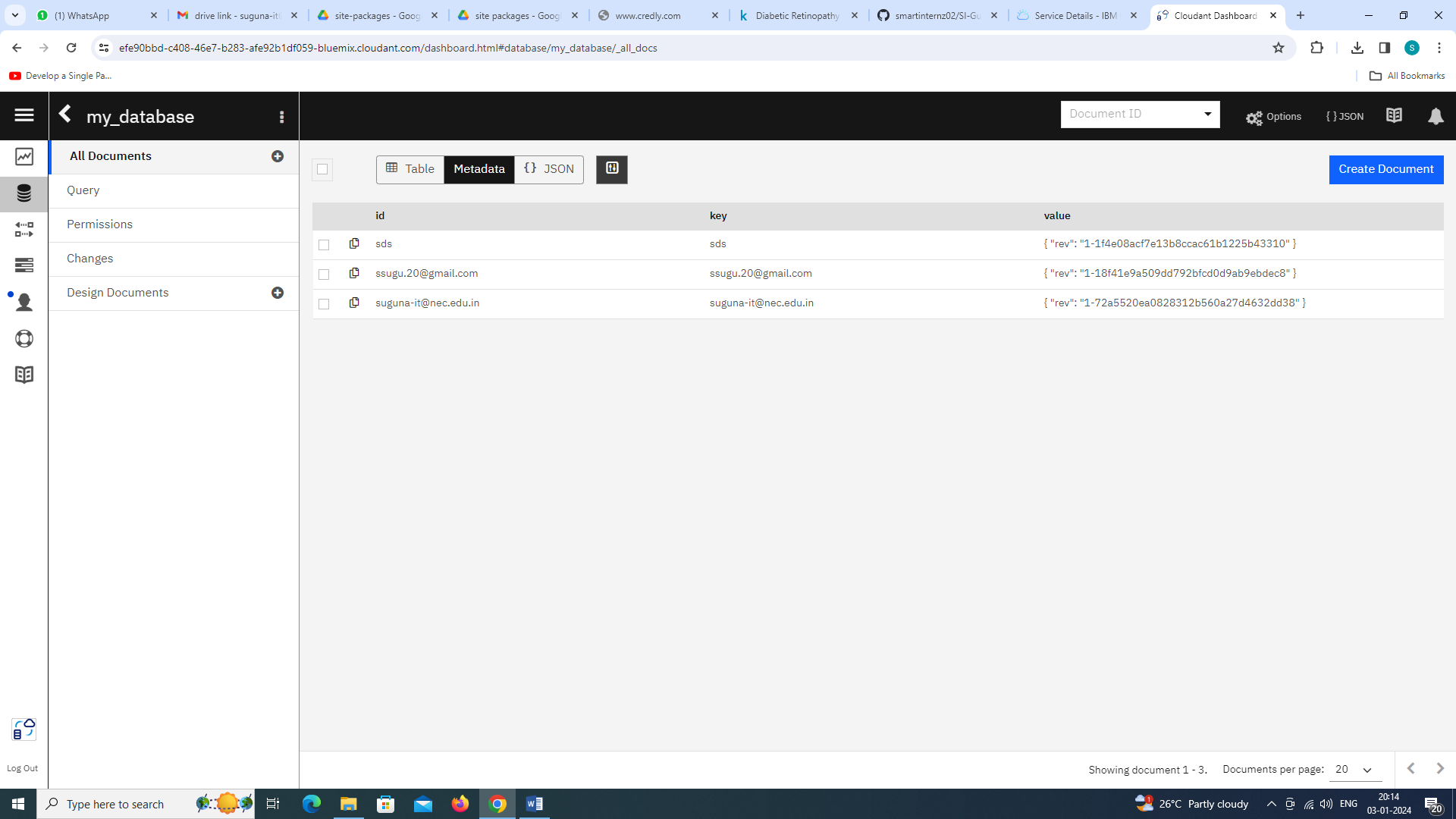






1. Cloudant – Database connection





**6. ADVANTAGES & DISADVANTAGES**

**6.1 Advantages**

**Scalability:** Cloud computing provides on-demand resources that can scale horizontally to handle varying workloads, making it well-suited for the resource-intensive nature of AI/ML tasks.

**Cost-Efficiency:** Pay-as-you-go models in cloud computing allow organizations to optimize costs by paying only for the resources consumed, reducing upfront capital investments.

**Accessibility:** Cloud-based AI/ML services can be accessed from anywhere with an internet connection, facilitating collaboration and enabling remote teams to work on shared projects.

**Global Reach:** Cloud providers have data centers distributed globally, allowing organizations to deploy AI/ML applications closer to end-users, reducing latency and improving user experience.

**Managed Services:** Cloud platforms offer managed AI/ML services that abstract the complexities of infrastructure management, enabling organizations to focus on model development and deployment rather than infrastructure maintenance.

**Innovation Acceleration:** Cloud computing environments provide a flexible and agile infrastructure, allowing organizations to experiment, innovate, and rapidly prototype new AI/ML models without significant upfront investments.

**Collaboration:** Cloud-based platforms facilitate collaborative development by providing centralized environments for model development, testing, and deployment, fostering teamwork among data scientists and engineers.

**Security Measures:** Leading cloud providers invest heavily in security measures, including data encryption, access controls, and compliance certifications, ensuring a secure environment for AI/ML applications.

**6.2 Disadvantages**

**Data Privacy Concerns:** Storing sensitive data in the cloud raises concerns about data privacy and security, especially when dealing with regulatory compliance and industry-specific requirements.

**Latency Issues:** The reliance on cloud infrastructure may introduce latency, which can be a critical factor for real-time AI/ML applications, such as those deployed in edge computing scenarios.

**Dependency on Internet Connectivity:** Cloud-based solutions require a reliable internet connection. In situations where connectivity is unstable or unavailable, access to AI/ML services may be disrupted.

**Cost Variability:** While cloud computing can be cost-efficient, costs can become unpredictable as workloads fluctuate, and data storage requirements increase, potentially leading to unexpected expenses.

**Limited Control Over Infrastructure:** Organizations have limited control over the underlying infrastructure in cloud environments. This lack of control may be a concern for those who require fine-tuned optimization for specific hardware configurations.

**Vendor Lock-in:** Adopting specific cloud services may lead to vendor lock-in, making it challenging to switch providers due to dependencies on proprietary APIs and services.

**Security Risks:** While cloud providers implement robust security measures, the shared nature of cloud environments introduces potential security risks, such as unauthorized access and data breaches.

**Regulatory Compliance Challenges:** Compliance with data protection regulations and industry-specific standards may pose challenges, particularly when data is stored or processed across different geographical regions.

**Learning Curve:** Transitioning to cloud-based AI/ML solutions may require organizations to invest in training and upskilling their workforce to adapt to new tools and platforms.

**Environmental Impact:** The energy consumption of large-scale cloud data centers can contribute to environmental concerns. Balancing the growing demand for AI/ML services with sustainable computing practices is a challenge.

**7. CONCLUSION**

In conclusion, the integration of Artificial Intelligence (AI) and Machine Learning (ML) with cloud computing is a transformative force shaping modern technology. This symbiotic relationship leverages scalable computing resources, advanced data analytics, and intelligent decision-making capabilities. The synergy between AI/ML and cloud computing spans diverse industries, including healthcare, finance, manufacturing, and agriculture. The advantages include scalability, cost-efficiency, accessibility, and global reach, enabling organizations to innovate without substantial upfront investments. Managed services simplify AI/ML development, allowing a focus on problem-solving rather than infrastructure management. However, challenges like data privacy concerns and potential vendor lock-in must be carefully navigated. Addressing environmental impacts and promoting sustainable computing practices are evolving considerations. Proposed systems, such as enhanced AutoML with federated learning and serverless AI/ML deployment frameworks, demonstrate ongoing efforts to refine and advance this integration. These innovations aim to overcome limitations and contribute to the evolution of AI/ML and cloud computing technologies. In summary, this integration continues to drive efficiency, informed decision-making, and new possibilities across science, business, and society.

**8. FUTURE SCOPE**

The future scope of integrating Artificial Intelligence (AI) and Machine Learning (ML) with cloud computing is exceptionally promising, with numerous opportunities for advancements and innovations. Here are key areas of future scope:

**Advanced AI Algorithms:**

The development of more sophisticated AI algorithms, including deep learning models, will continue. Enhanced algorithms will lead to better accuracy, faster processing, and improved decision-making capabilities.

**Edge Computing Integration:**

The integration of AI/ML with edge computing is expected to grow. This approach will enable real-time processing of data closer to the source, reducing latency and enhancing the performance of applications in fields like healthcare, manufacturing, and IoT.

**Explainable AI (XAI):**

Future AI systems will likely prioritize explainability, making it easier for users to understand and trust the decisions made by AI models. Explainable AI is crucial, especially in sensitive domains such as healthcare and finance.

**Quantum Computing and AI:**

The intersection of quantum computing and AI holds significant potential. Quantum computing may revolutionize the processing power available for complex AI tasks, opening new frontiers in machine learning capabilities.

**AutoML Advancements:**

AutoML tools are likely to become more sophisticated, automating not only model selection and hyperparameter tuning but also addressing feature engineering and data preprocessing, making AI more accessible to non-experts.

**REFERENCES**

1. Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA, 316(22), 2402-2410.
2. Abràmoff, M. D., Lou, Y., Erginay, A., Clarida, W., Amelon, R., Folk, J. C., ... & Niemeijer, M. (2016). Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning. Investigative ophthalmology & visual science, 57(13), 5200-5206
3. Ting, D. S. W., Cheung, C. Y. L., Lim, G., Tan, G. S. W., Quang, N. D., Gan, A., ... & Wong, T. Y. (2017). Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. JAMA, 318(22), 2211-2223
4. Gargeya, R., & Leng, T. (2017). Automated identification of diabetic retinopathy using deep learning. Ophthalmology, 124(7), 962-969
5. Li, Z., Keel, S., Liu, C., He, Y., Meng, W., & Scheetz, J. (2017). An automated grading system for detection of vision-threatening referable diabetic retinopathy on the basis of color fundus photographs. Diabetes care, 40(3), 365-371