# MediScan AI APP : AN AI DRIVEN MEDICAL DIAGNOSIS ASSISTANT

**13-November 2024**

**By**

**Mukhilan A M**

# STEP 1 - PROTOTYPE SELECTION :

## Abstract :

In recent years, the integration of Artificial Intelligence (AI) into healthcare has significantly improved diagnostic accuracy and efficiency. This project focuses on developing an innovative mobile application that utilizes AI-driven algorithms to diagnose diseases based on medical images such as X-rays, MRIs, and CT scans. The primary goal is to assist healthcare professionals by offering rapid, reliable, and accessible diagnostic insights, especially in environments with limited medical resources.

The app leverages deep learning techniques, specifically convolutional neural networks (CNNs), which are trained on large datasets of labeled medical images. These models are capable of identifying and classifying patterns indicative of various diseases, such as pneumonia, tumors, fractures, or other abnormalities. With its user friendly interface, medical professionals or users can upload images, receive a diagnosis within seconds, and access detailed reports that explain the AI's findings. The app is designed to be simple and easy to use for everyone, whether you're a doctor or someone looking for medical advice. The interface is clean and straightforward, making it easy to upload medical images and get results without any confusion.

Additionally, the app incorporates continuous learning capabilities, allowing it to improve its accuracy over time as more data is processed. Ensuring patient data privacy and adhering to medical data standards are key priorities in the design of this system. By bridging the gap between advanced AI technology and healthcare diagnostics, this app aims to democratize healthcare access and reduce the burden on medical professionals through faster, data-driven diagnoses.

- **Feasibility**: The project leverages current AI and web technologies, making development feasible in the short term.
- **Viability**: Medical diagnostics and healthcare technology will likely remain crucial for decades, supporting long-term relevance.
- **Monetization**: The product could be monetized directly through subscription models, pay-per-analysis, or partnerships with healthcare providers.

# STEP 2 - PROTOTYPE DEVELOPMENT :

The main motivation behind this project is to create a Web App platform for medical image analysis across various types of medical images.I do this by Creating unique Convolutional Neural Network (CNN) models tailored to different types of medical Images, where each type of Medical Image will have a unique model corresponding to them.

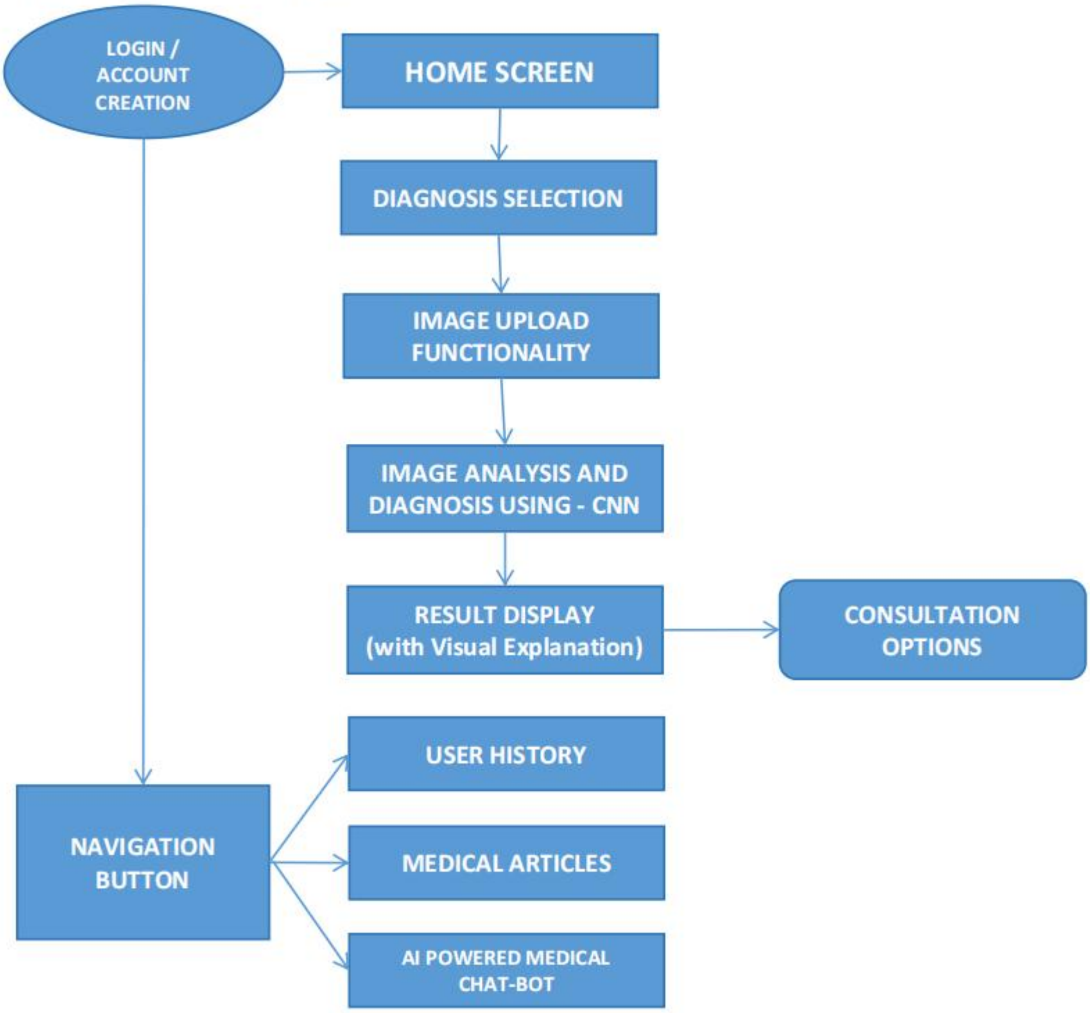
Large datasets are gathered for each type of medical imaging, pre-processed, and data augmented by creating new images from pre-existing ones.There are two types of data: testing and training.

When given photos as input in the form of a matrix, each CNN model is specifically created to extract pertinent features from the associated type of medical image. With these images, feature extraction is carried out by applying several Conv2D layers for convolutional processes where The Conv2D layer uses convolutional filters to extract local patterns and characteristics from the input image. Each filter learns to identify unique patterns, like edges, textures, or forms, by convolving over the input image. By lowering spatial dimensions while keeping the most crucial properties, maxpooling2D layers enable downsampling.In doing so, it helps to abstract and summarize the extracted characteristics, increasing the model's invariance to slight spatial fluctuations and lowering computational cost. It accomplishes this by choosing the maximum value within each pooling window. After which the Matrix is flattened and Dense layers are incorporated to produce the final diagnosis output as they receive the flattened feature vector as input and learn to associate these features with specific classes or labels during the training process.Following model construction, the models are fitted, trained using the training dataset, and fine-tuned to maximize accuracy and performance in CNN models. Our objective is to optimize model performance and provide reliable diagnosis outcomes.

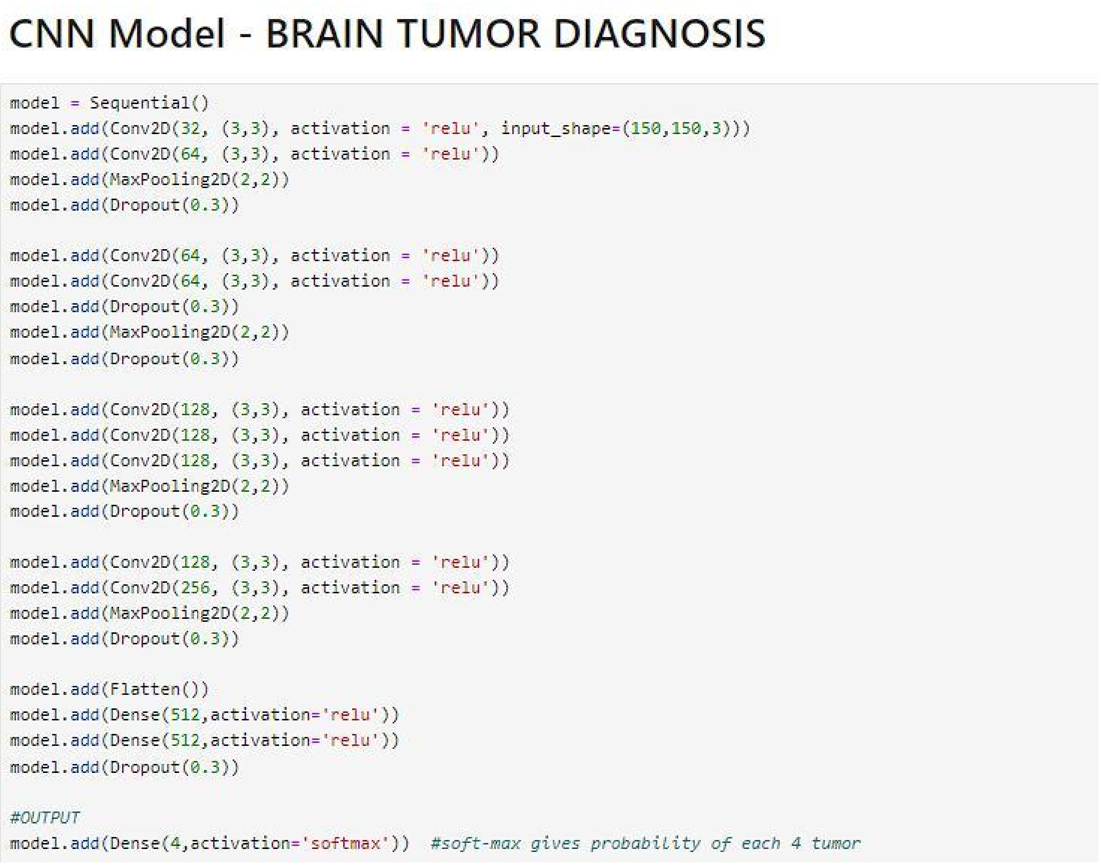
After Models are created for different Medical Image Types, Streamlit is used to integrate these trained CNN models into a web application allows for smooth end-user interaction. From the user interface, users can choose the desired sort of medical diagnosis, such as the detection of pneumonia, skin cancer, brain tumors, or fractures. They can then upload the matching medical image as input after that. The front-end interface utilizes the unique CNN model created for that particular Medical Image type selected by the user to perform real-time diagnosis. Leveraging the power of deep learning, the web application provides the diagnosis results, empowering healthcare professionals and individuals alike in making informed decisions regarding patient care and treatment planning.

Overall, this project aims to democratize medical image analysis by harnessing the capabilities of CNNs and web technologies, ultimately improving healthcare outcomes and patient well-being.

**PROTOTYPE DIAGRAM:**

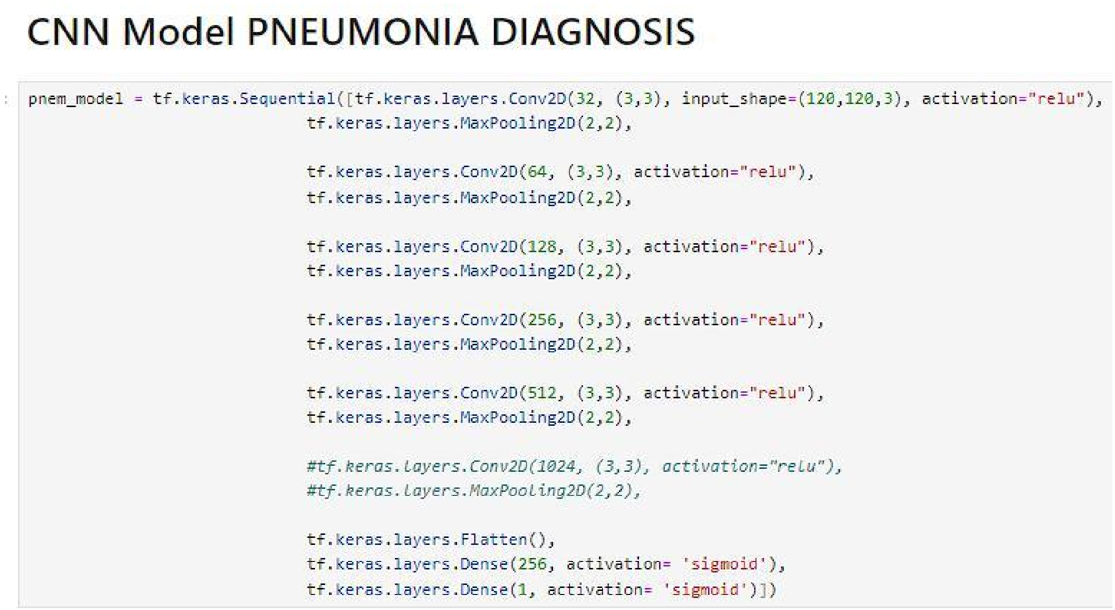

LOGIN / ACCOUNT CREATION
HOME SCREEN
DIAGNOSIS SELECTION
IMAGE UPLOAD FUNCTIONALITY
IMAGE ANALYSIS AND DIAGNOSIS USING - CNN
RESULT DISPLAY (with Visual Explanation)
CONSULTATION OPTIONS
USER HISTORY
NAVIGATION BUTTON
MEDICAL ARTICLES
AI POWERED MEDICAL CHAT-BOT

# Code Implementation Snippets :



## CNN Model - BRAIN TUMOR DIAGNOSIS

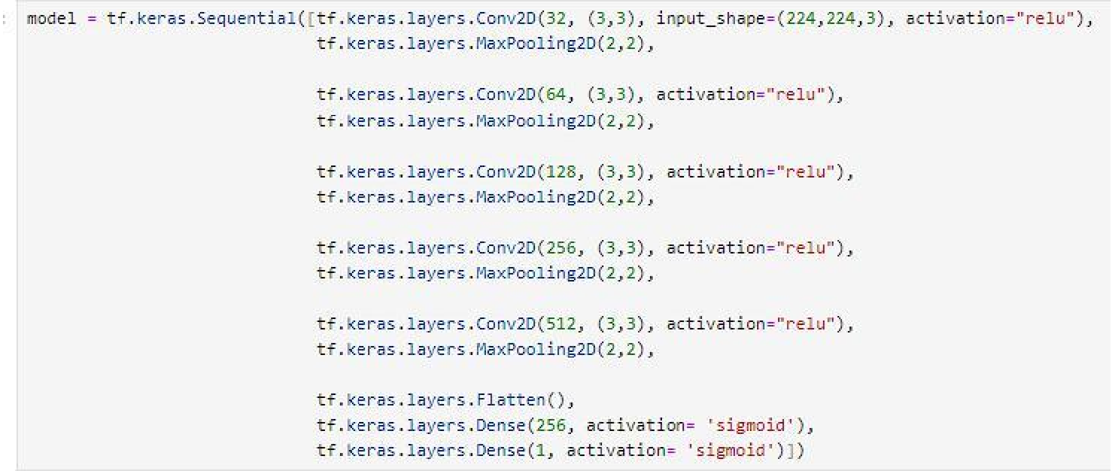
```
model = Sequential()
model.add(Conv2D(32, (3,3), activation = 'relu', input_shape=(150,150,3)))
model.add(Conv2D(64, (3,3), activation = 'relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))

model.add(Conv2D(64, (3,3), activation = 'relu'))
model.add(Conv2D(64, (3,3), activation = 'relu'))
model.add(Dropout(0.3))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))

model.add(Conv2D(128, (3,3), activation = 'relu'))
model.add(Conv2D(128, (3,3), activation = 'relu'))
model.add(Conv2D(128, (3,3), activation = 'relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))

model.add(Conv2D(128, (3,3), activation = 'relu'))
model.add(Conv2D(256, (3,3), activation = 'relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.3))

#OUTPUT
model.add(Dense(4,activation='softmax'))  #soft-max gives probability of each 4 tumor
```



## CNN Model PNEUMONIA DIAGNOSIS

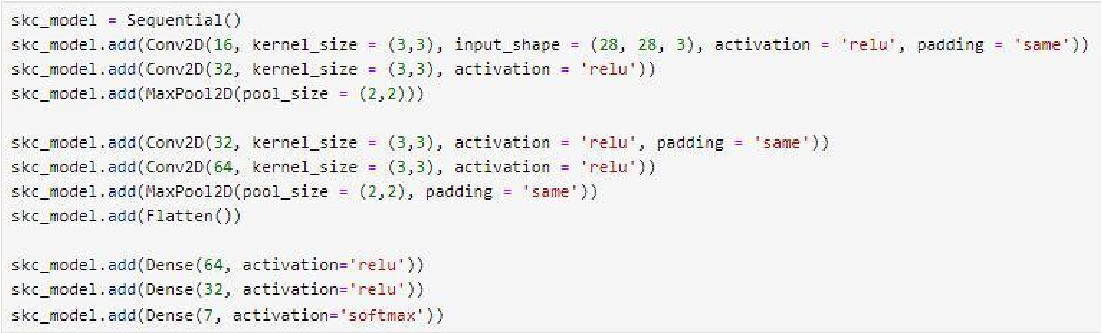
```
pnem_model = tf.keras.Sequential([tf.keras.layers.Conv2D(32, (3,3), input_shape=(120,120,3), activation="relu"),
                            tf.keras.layers.MaxPooling2D(2,2),

                            tf.keras.layers.Conv2D(64, (3,3), activation="relu"),
                            tf.keras.layers.MaxPooling2D(2,2),

                            tf.keras.layers.Conv2D(128, (3,3), activation="relu"),
                            tf.keras.layers.MaxPooling2D(2,2),

                            tf.keras.layers.Conv2D(256, (3,3), activation="relu"),
                            tf.keras.layers.MaxPooling2D(2,2),

                            tf.keras.layers.Conv2D(512, (3,3), activation="relu"),
                            tf.keras.layers.MaxPooling2D(2,2),

                            #tf.keras.layers.Conv2D(1024, (3,3), activation="relu"),
                            #tf.keras.layers.MaxPooling2D(2,2),

                            tf.keras.layers.Flatten(),
                            tf.keras.layers.Dense(256, activation= 'sigmoid'),
                            tf.keras.layers.Dense(1, activation= 'sigmoid')])
```

## CNN Model FRACTURE DIAGNOSIS



```
model = tf.keras.Sequential([tf.keras.layers.Conv2D(32, (3,3), input_shape=(224,224,3), activation="relu"),
                             tf.keras.layers.MaxPooling2D(2,2),

                             tf.keras.layers.Conv2D(64, (3,3), activation="relu"),
                             tf.keras.layers.MaxPooling2D(2,2),

                             tf.keras.layers.Conv2D(128, (3,3), activation="relu"),
                             tf.keras.layers.MaxPooling2D(2,2),

                             tf.keras.layers.Conv2D(256, (3,3), activation="relu"),
                             tf.keras.layers.MaxPooling2D(2,2),

                             tf.keras.layers.Conv2D(512, (3,3), activation="relu"),
                             tf.keras.layers.MaxPooling2D(2,2),

                             tf.keras.layers.Flatten(),
                             tf.keras.layers.Dense(256, activation= 'sigmoid'),
                             tf.keras.layers.Dense(1, activation= 'sigmoid')])
```

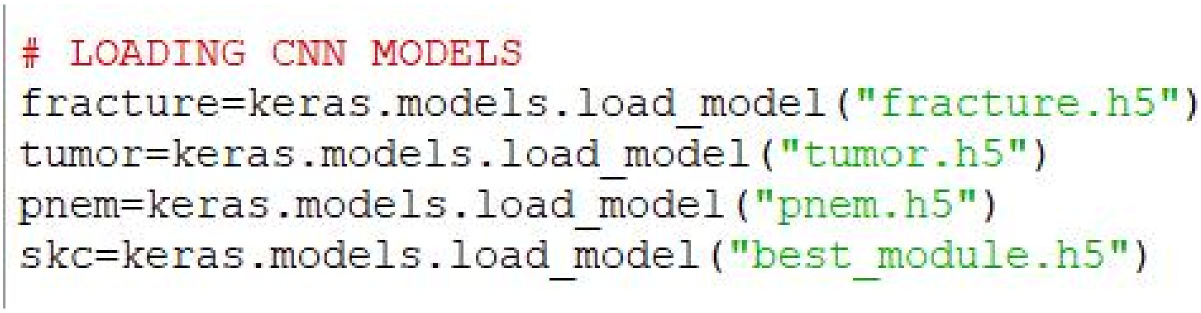
## CNN Model SKIN CANCER DETECTION



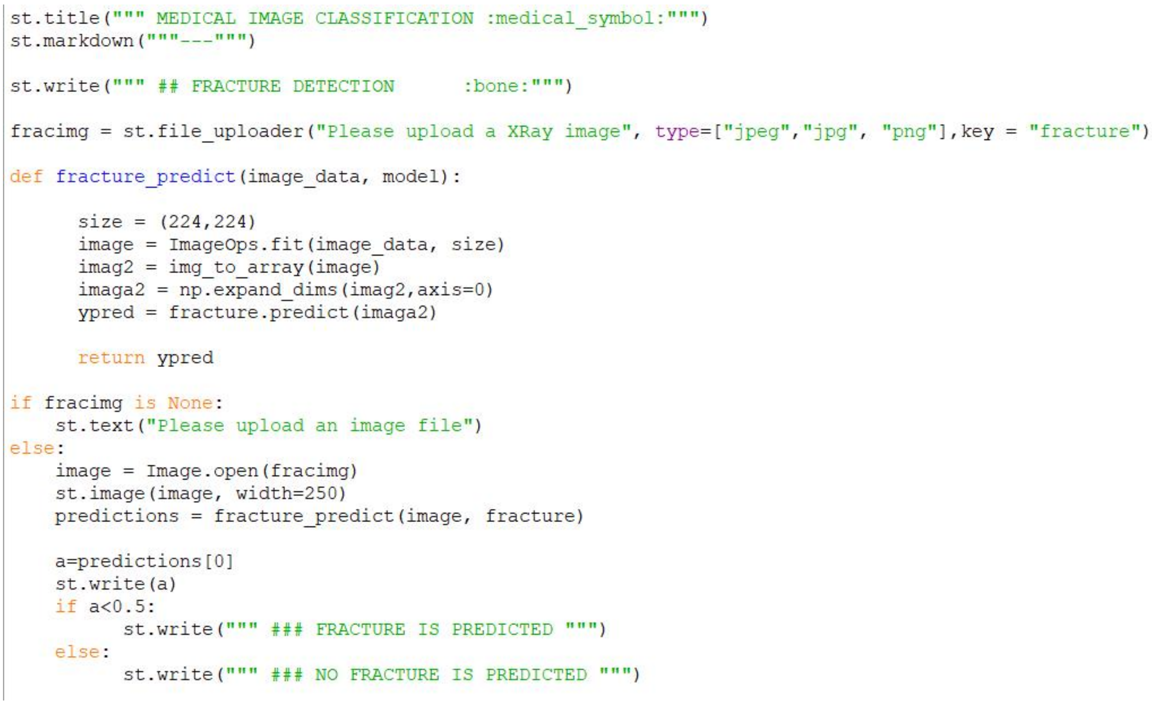
```
skc_model = Sequential()
skc_model.add(Conv2D(16, kernel_size = (3,3), input_shape = (28, 28, 3), activation = 'relu', padding = 'same'))
skc_model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu'))
skc_model.add(MaxPool2D(pool_size = (2,2)))

skc_model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu', padding = 'same'))
skc_model.add(Conv2D(64, kernel_size = (3,3), activation = 'relu'))
skc_model.add(MaxPool2D(pool_size = (2,2), padding = 'same'))
skc_model.add(Flatten())

skc_model.add(Dense(64, activation='relu'))
skc_model.add(Dense(32, activation='relu'))
skc_model.add(Dense(7, activation='softmax'))
```

The above CNN Models will be compiled and then trained and saved. The saved models will be used to create webapp

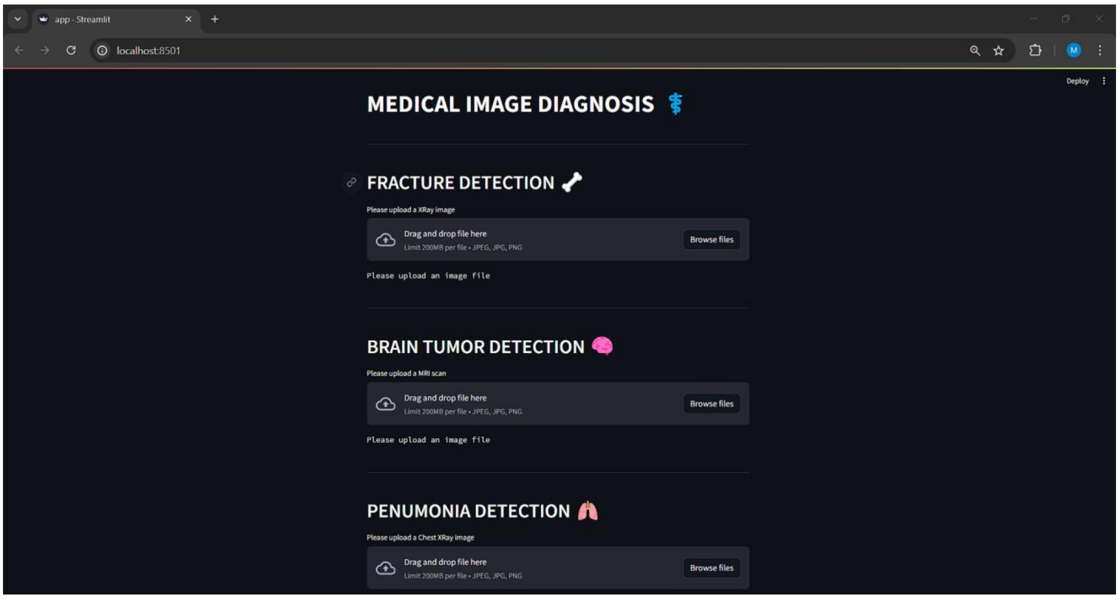
## WEBAPP (streamlit) -



```
# LOADING CNN MODELS
fracture=keras.models.load_model("fracture.h5")
tumor=keras.models.load_model("tumor.h5")
pnem=keras.models.load_model("pnem.h5")
skc=keras.models.load_model("best_module.h5")
```



```
st.title(""" MEDICAL IMAGE CLASSIFICATION :medical_symbol:""")
st.markdown("""---""")

st.write(""" ## FRACTURE DETECTION      :bone:""")

fracimg = st.file_uploader("Please upload a XRay image", type=["jpeg","jpg", "png"],key = "fracture")

def fracture_predict(image_data, model):

      size = (224,224)
      image = ImageOps.fit(image_data, size)
      imag2 = img_to_array(image)
      imaga2 = np.expand_dims(imag2,axis=0)
      ypred = fracture.predict(imaga2)

      return ypred

if fracimg is None:
    st.text("Please upload an image file")
else:
    image = Image.open(fracimg)
    st.image(image, width=250)
    predictions = fracture_predict(image, fracture)

    a=predictions[0]
    st.write(a)
    if a<0.5:
          st.write(""" ### FRACTURE IS PREDICTED """)
    else:
          st.write(""" ### NO FRACTURE IS PREDICTED """)
```



MEDICAL IMAGE DIAGNOSIS ⚕

FRACTURE DETECTION 🦴

Please upload a XRay image

```
Please upload an image file
```

BRAIN TUMOR DETECTION 🧠

Please upload a MRI scan

```
Please upload an image file
```

PENUMONIA DETECTION 🫁

Please upload a Chest XRay image

GITHUB :
https://github.com/Mukhilan22/Medical-Image-Analysis-CNN

MEDICAL IMAGE ANALYSIS.ipynb - Model Building
App.py - Web App

# STEP 3 - BUSINESS MODELING :

## Value Proposition:

**Core Offer**: A web-based medical image analysis tool using CNN models for diagnostic support on various medical image types (e.g., X-rays, MRIs, CT scans).

**Benefit**: Quick and accessible diagnostic insights to support healthcare professionals and empower patients in underserved or rural areas.

**Differentiation**: Tailored CNN models for each image type, increasing accuracy and reliability in diverse diagnostic applications, and a user-friendly interface using Streamlit.

## Customer Segments:

**Primary Users**:

Healthcare providers, clinics, and hospitals looking for a quick diagnostic support tool.

Independent radiologists and diagnostic centers needing a supplementary tool for faster image analysis.

**Secondary Users**:

Research institutions focusing on medical imaging and AI.

Direct-to-consumer option for patients seeking a preliminary diagnosis (where permissible).

## Revenue Streams:

**Subscription Model**: Offer tiered subscription plans for clinics and hospitals based on the volume of analyses needed per month.

**Pay-per-use**: Charge for each image analysis if users don’t want a subscription.

**B2B Partnerships**: Collaborate with large healthcare providers and hospitals on long-term contracts for exclusive diagnostic support.

**Data Licensing (optional)**: If data is collected ethically and with permission, it could be anonymized and licensed to researchers or universities for training other AI models or research.

## Channels:

**Direct Online Access**: Website as the main portal for accessing the web app.

**Healthcare Technology Partners**: Collaborate with healthcare tech platforms or electronic health record (EHR) providers to integrate the app for direct access by healthcare facilities.

**Conferences and Medical AI Events**: Engage with the healthcare community at relevant industry events to promote the platform.

**Online Marketing**: Use social media, medical journals, and targeted online ads to reach healthcare professionals.

## Customer Relationships:

**Customer Support**: Provide dedicated support for hospitals and clinics, including troubleshooting, model updates, and training sessions on new features.

**Training and Onboarding**: Offer initial training to ensure customers are comfortable using the platform.

**Feedback Loop**: Regular feedback collection to adapt and improve the models and user interface based on user input.

## Key Resources:

**Data and Labeling Resources**: A large, high-quality dataset for model training and refinement.

**AI/ML Expertise**: Skilled AI professionals to continuously improve and update the CNN models.

**Cloud Infrastructure**: Servers and storage for data processing, model deployment, and app hosting.

**Web Development**: Expertise to build and maintain the front-end and back-end of the Streamlit web app.

## Key Partners:

**Hospitals and Diagnostic Centers**: Early adopters that can provide data and feedback for model training.

**Healthcare IT Companies**: Collaborate for integrated solutions and API connectivity with EHRs.

**Academic and Research Institutions**: Collaborate on model validation, testing, and potential research publications.

## Cost Structure:

**Model Development and Maintenance**: Costs associated with developing, training, and updating CNN models.

**Cloud Hosting**: Costs for server space, storage, and computational resources.

**Marketing and Sales**: Expenses for promoting the platform and acquiring customers.

**Customer Support**: Ongoing expenses for customer assistance and maintenance.

## Example Revenue Projections and Scalability

With a subscription or pay-per-use model, project potential revenues based on user growth and pricing. For instance, charging $200/month for clinics or $10 per analysis could scale as more healthcare providers adopt the technology.

# STEP 4 - FINANCIAL MODELING (EQUATION WITH DATA ANALYSIS AND MACHINE LEARNING :

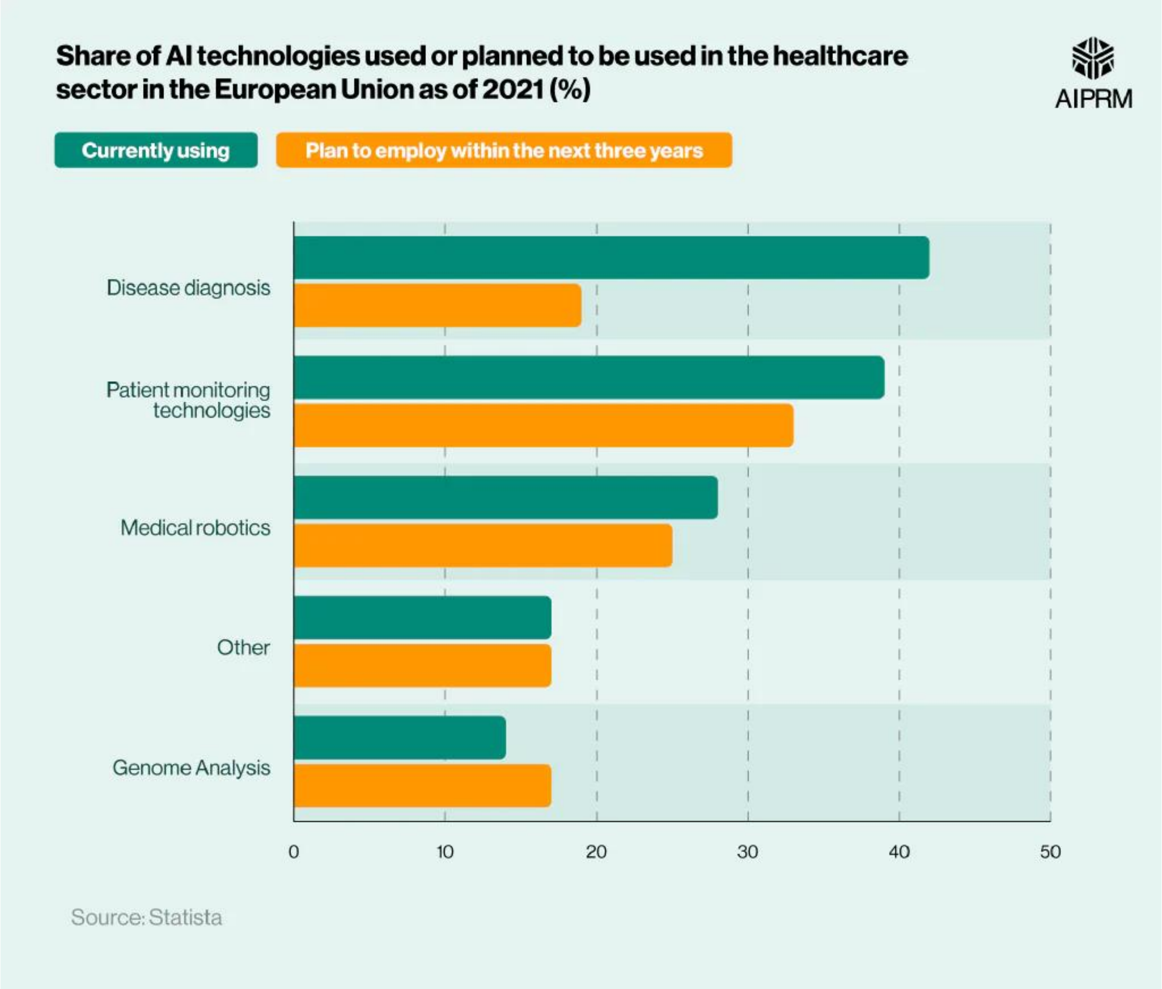
## a. Identify which Market your product/service will be launched into

For a medical image analysis platform powered by CNN models, your target market falls within **Healthcare and Medical Diagnostics** with a focus on **AI-driven Diagnostic Tools**. This market includes segments such as:

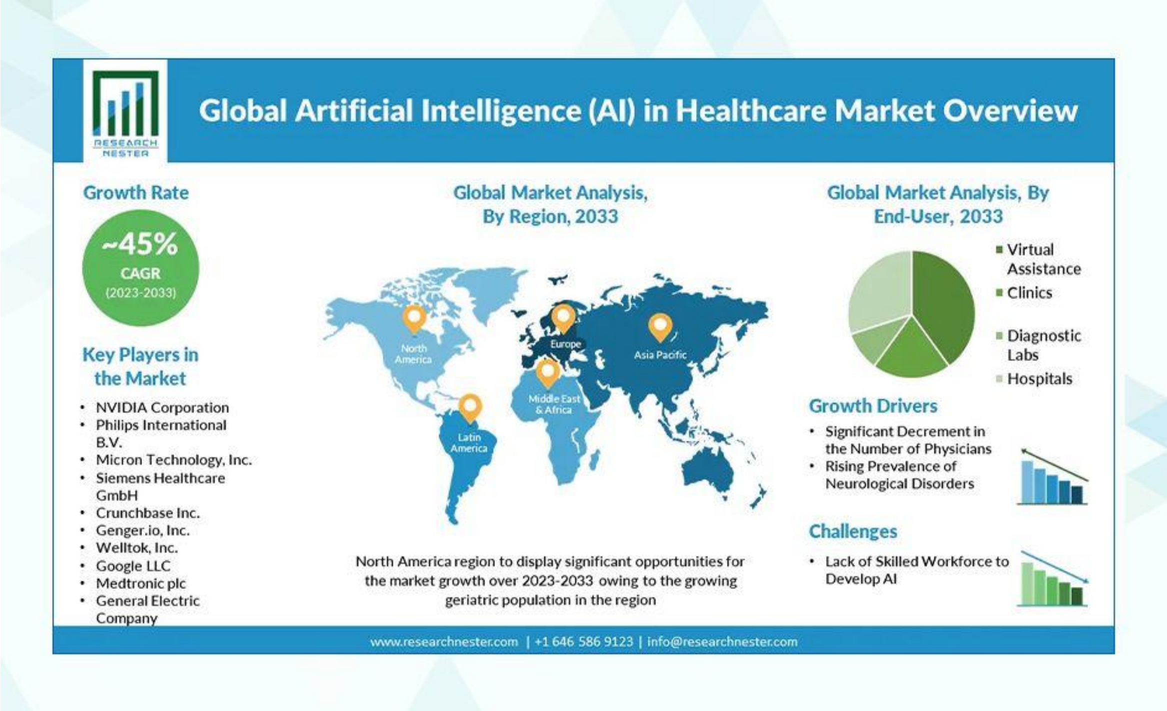
1. **Hospitals and Clinics**: Facilities looking to improve diagnostic speed and accuracy through AI.
2. **Diagnostic and Radiology Centers**: Organizations aiming to streamline image analysis for various medical conditions.
3. **Telemedicine Providers**: Companies offering remote diagnostic services that could use AI-assisted tools for virtual consultations.
4. **Individual Healthcare Practitioners**: Radiologists and independent physicians who may use AI as a secondary check for their diagnoses.
5. **Direct-to-Consumer (where regulations permit)**: Individuals seeking preliminary diagnoses before visiting a healthcare provider.
6. **Research Institutions and Universities**: Entities focusing on medical imaging and AI research, which might leverage the platform for research or educational purposes.

Given this information, we can outline a preliminary financial model that predicts revenue streams based on different types of customer segments and usage patterns.

## b. Collect Data/Statistics Regarding that Market Online :



Source: Statista



### Global Artificial Intelligence (AI) in Healthcare Market Overview

RESEARCH NESTER

**Growth Rate**

~45% CAGR (2023-2033)

**Key Players in the Market**

- NVIDIA Corporation
- Philips International B.V.
- Micron Technology, Inc.
- Siemens Healthcare GmbH
- Crunchbase Inc.
- Genger.io, Inc.
- Welltok, Inc.
- Google LLC
- Medtronic plc
- General Electric Company

**Global Market Analysis, By Region, 2033**

North America
Europe
Asia Pacific
Middle East & Africa
Latin America

North America region to display significant opportunities for the market growth over 2023-2033 owing to the growing geriatric population in the region

**Global Market Analysis, By End-User, 2033**

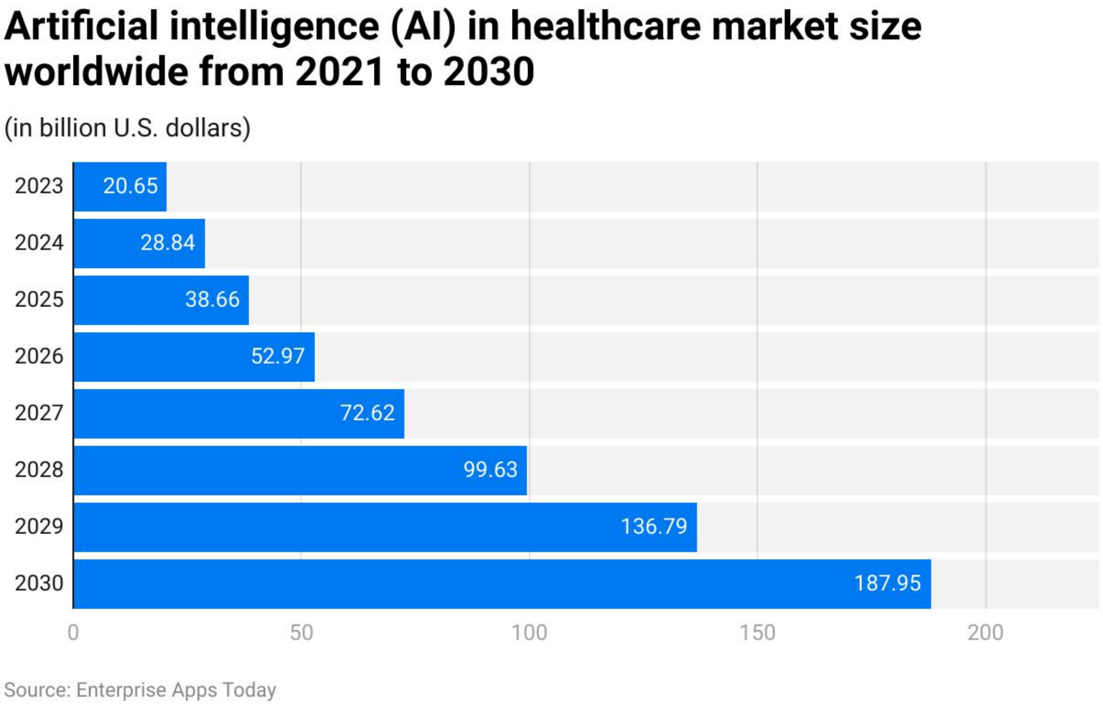
**Growth Drivers**

- Significant Decrement in the Number of Physicians
- Rising Prevalence of Neurological Disorders

**Challenges**

- Lack of Skilled Workforce to Develop AI

www.researchnester.com | +1 646 586 9123 | info@researchnester.com

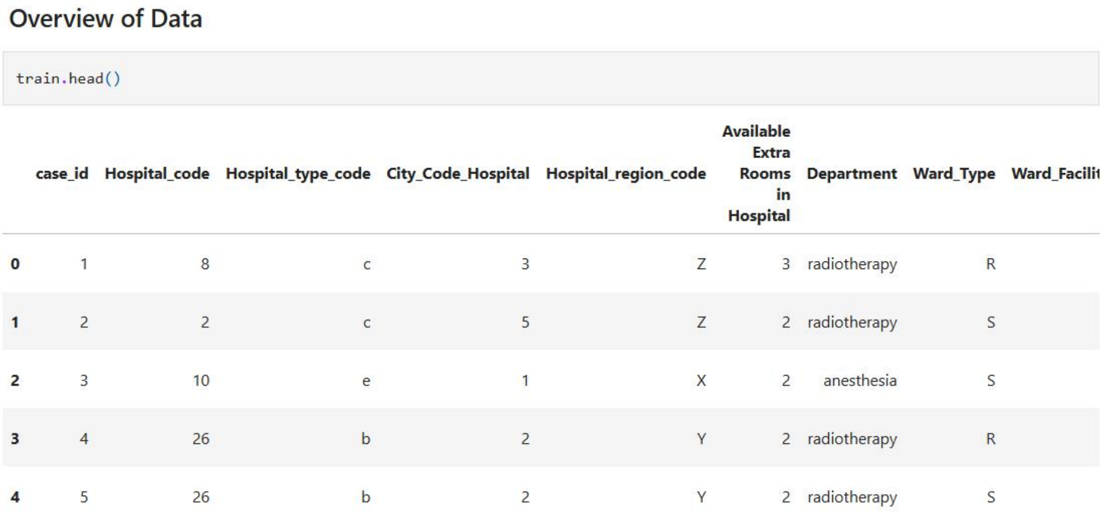


Artificial intelligence (AI) in healthcare market size worldwide from 2021 to 2030

(in billion U.S. dollars)

Source: Enterprise Apps Today

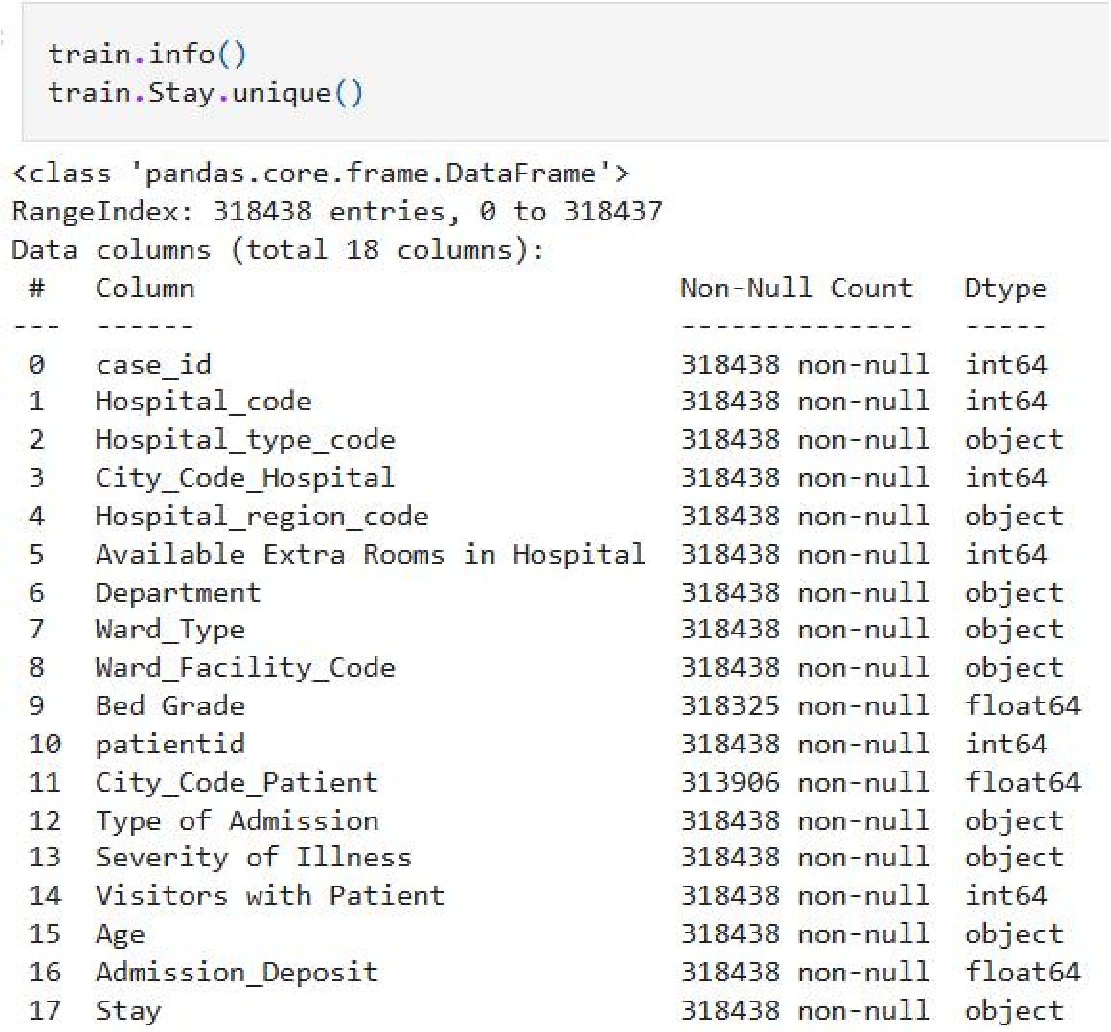
## c. Perform forcast/prediction on the market using regression models or time series forcasting :



Overview of Data

```
train.head()
```

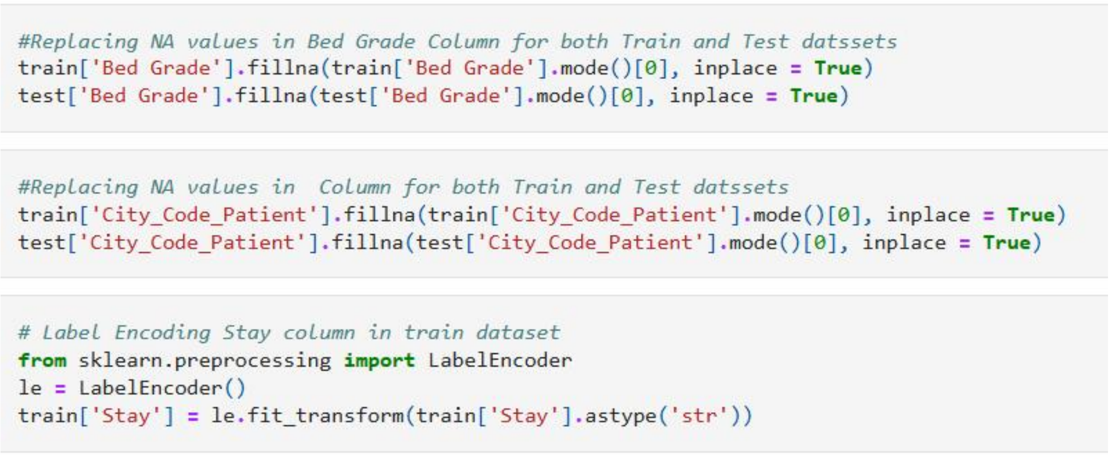
| | case_id | Hospital_code | Hospital_type_code | City_Code_Hospital | Hospital_region_code | Available Extra Rooms in Hospital | Department | Ward_Type | Ward_Facilit |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 8 | c | 3 | Z | 3 | radiotherapy | R | |
| **1** | 2 | 2 | c | 5 | Z | 2 | radiotherapy | S | |
| **2** | 3 | 10 | e | 1 | X | 2 | anesthesia | S | |
| **3** | 4 | 26 | b | 2 | Y | 2 | radiotherapy | R | |
| **4** | 5 | 26 | b | 2 | Y | 2 | radiotherapy | S | |



```
train.info()
train.Stay.unique()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 318438 entries, 0 to 318437
Data columns (total 18 columns):
 #   Column                             Non-Null Count   Dtype  
---  ------                             --------------   -----  
 0   case_id                            318438 non-null  int64  
 1   Hospital_code                      318438 non-null  int64  
 2   Hospital_type_code                 318438 non-null  object 
 3   City_Code_Hospital                 318438 non-null  int64  
 4   Hospital_region_code               318438 non-null  object 
 5   Available Extra Rooms in Hospital  318438 non-null  int64  
 6   Department                         318438 non-null  object 
 7   Ward_Type                          318438 non-null  object 
 8   Ward_Facility_Code                 318438 non-null  object 
 9   Bed Grade                          318325 non-null  float64
 10  patientid                          318438 non-null  int64  
 11  City_Code_Patient                  313906 non-null  float64
 12  Type of Admission                  318438 non-null  object 
 13  Severity of Illness                318438 non-null  object 
 14  Visitors with Patient              318438 non-null  int64  
 15  Age                                318438 non-null  object 
 16  Admission_Deposit                  318438 non-null  float64
 17  Stay                               318438 non-null  object 
```

## Data Preparation



```
#Replacing NA values in Bed Grade Column for both Train and Test datssets
train['Bed Grade'].fillna(train['Bed Grade'].mode()[0], inplace = True)
test['Bed Grade'].fillna(test['Bed Grade'].mode()[0], inplace = True)
```

```
#Replacing NA values in  Column for both Train and Test datssets
train['City_Code_Patient'].fillna(train['City_Code_Patient'].mode()[0], inplace = True)
test['City_Code_Patient'].fillna(test['City_Code_Patient'].mode()[0], inplace = True)
```

```
# Label Encoding Stay column in train dataset
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train['Stay'] = le.fit_transform(train['Stay'].astype('str'))
```

## Feature Engineering

```
def get_countid_enocde(train, test, cols, name):
  temp = train.groupby(cols)['case_id'].count().reset_index().rename(columns = {'case_id': name})
  temp2 = test.groupby(cols)['case_id'].count().reset_index().rename(columns = {'case_id': name})
  train = pd.merge(train, temp, how='left', on= cols)
  test = pd.merge(test,temp2, how='left', on= cols)
  train[name] = train[name].astype('float')
  test[name] = test[name].astype('float')
  train[name].fillna(np.median(temp[name]), inplace = True)
  test[name].fillna(np.median(temp2[name]), inplace = True)
  return train, test
```

```
train, test = get_countid_enocde(train, test, ['patientid'], name = 'count_id_patient')
train, test = get_countid_enocde(train, test,
                                 ['patientid', 'Hospital_region_code'], name = 'count_id_patient_hospitalCode')
train, test = get_countid_enocde(train, test,
                                 ['patientid', 'Ward_Facility_Code'], name = 'count_id_patient_wardfacilityCode')
```

```
# Droping duplicate columns
test1 = test.drop(['Stay', 'patientid', 'Hospital_region_code', 'Ward_Facility_Code'], axis =1)
train1 = train.drop(['case_id', 'patientid', 'Hospital_region_code', 'Ward_Facility_Code'], axis =1)
```

```
# Splitting train data for Naive Bayes and XGBoost
X1 = train1.drop('Stay', axis =1)
y1 = train1['Stay']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size =0.20, random_state =100)
```

# Models

## Naive Bayes Model

```
from sklearn.naive_bayes import GaussianNB
target = y_train.values
features = X_train.values
classifier_nb = GaussianNB()
model_nb = classifier_nb.fit(features, target)
```

```
prediction_nb = model_nb.predict(X_test)
from sklearn.metrics import accuracy_score
acc_score_nb = accuracy_score(prediction_nb,y_test)
print("Acurracy:", acc_score_nb*100)
```

```
Acurracy: 34.55439015199096
```

## XGBoost Model

```
import xgboost
classifier_xgb = xgboost.XGBClassifier(max_depth=4, learning_rate=0.1, n_estimators=800,
                                   objective='multi:softmax', reg_alpha=0.5, reg_lambda=1.5,
                                   booster='gbtree', n_jobs=4, min_child_weight=2, base_score= 0.75)
```

```
model_xgb = classifier_xgb.fit(X_train, y_train)
```

```
prediction_xgb = model_xgb.predict(X_test)
acc_score_xgb = accuracy_score(prediction_xgb,y_test)
print("Accuracy:", acc_score_xgb*100)
```

## Predictions

```
# Naive Bayes
pred_nb = classifier_nb.predict(test1.iloc[:,1:])
result_nb = pd.DataFrame(pred_nb, columns=['Stay'])
result_nb['case_id'] = test1['case_id']
result_nb = result_nb[['case_id', 'Stay']]
```

```
result_nb['Stay'] = result_nb['Stay'].replace({0:'0-10', 1: '11-20', 2: '21-30', 3:'31-40', 4: '41-50', 5: '51-60', 6: '61-7
result_nb.head()
```

| | case_id | Stay |
|---|---|---|
| **0** | 318439 | 21-30 |
| **1** | 318440 | 51-60 |
| **2** | 318441 | 21-30 |
| **3** | 318442 | 21-30 |
| **4** | 318443 | 31-40 |

```
# XGBoost
pred_xgb = classifier_xgb.predict(test1.iloc[:,1:])
result_xgb = pd.DataFrame(pred_xgb, columns=['Stay'])
result_xgb['case_id'] = test1['case_id']
result_xgb = result_xgb[['case_id', 'Stay']]
```

```
result_xgb['Stay'] = result_xgb['Stay'].replace({0:'0-10', 1: '11-20', 2: '21-30', 3:'31-40', 4: '41-50', 5: '51-60', 6: '61
result_xgb.head()
```

| | case_id | Stay |
|---|---|---|
| **0** | 318439 | 0-10 |
| **1** | 318440 | 51-60 |
| **2** | 318441 | 21-30 |
| **3** | 318442 | 21-30 |
| **4** | 318443 | 51-60 |

## Results

```
# Naive Bayes
print(result_nb.groupby('Stay')['case_id'].nunique())
```

```
Stay
0-10                   2598
11-20                 26827
21-30                 72206
31-40                 15639
41-50                   469
51-60                 13651
61-70                    92
71-80                   955
81-90                   296
91-100                    2
More than 100 Days     4322
Name: case_id, dtype: int64
```

```
# XGBoost
print(result_xgb.groupby('Stay')['case_id'].nunique())
```

```
Stay
0-10                   4373
11-20                 39337
21-30                 58261
31-40                 12100
41-50                    61
51-60                 19217
61-70                    16
71-80                   302
81-90                  1099
91-100                   78
More than 100 Days     2213
Name: case_id, dtype: int64
```

## d. Design Financial Equation corresponding to that Market Trend :

## 1. Revenue Model Equations

**Monthly Subscription Revenue (MSR)**:

$$MSR=i=1\sum n \quad Si \quad \times Ps$$

Where:

- nnn = Number of subscribing organizations (e.g., hospitals, clinics).
- SiS_iSi = Subscription volume for each organization (e.g., number of users or devices).
- PsP_{s}Ps = Subscription price per unit (monthly fee per organization or user).

**Pay-Per-Use Revenue (PPUR)**:

$$PPUR=j=1\sum m \quad Tj \quad \times Pt$$

Where:

- mmm = Number of pay-per-use customers.
- TjT_jTj = Total images analyzed per customer (usage per customer per month).
- PtP_{t}Pt = Price per image analysis.

## 2. Total Monthly Revenue (TMR)

$$TMR=MSR+PPUR$$

## 3. Cost Structure Estimation

Estimate fixed and variable costs to arrive at profitability:

**Fixed Costs (FC)**: Costs that do not vary with customer usage, e.g., model development, cloud hosting, customer support.

$$FC=Cmodel \quad +Ccloud \quad +Csupport \quad +Cmarketing$$

Where:

- CmodelC_{model}Cmodel : Cost of initial model development.
- CcloudC_{cloud}Ccloud : Monthly cloud hosting and storage costs.

- CsupportC_{support}Csupport : Customer support and operations.
- CmarketingC_{marketing}Cmarketing : Monthly marketing expenses.

**Variable Costs (VC)**: Costs that vary with customer usage, such as computational costs for each image processed.

VC=j=1∑m Tj ×Cper_image

Where:
Cper_imageC_{per_image}Cper_image : Cost per image analysis (e.g., computation and storage).

## 4. Profitability Estimation (Monthly Profit)

Monthly Profit=TMR−(FC+VC)

This financial model provides a structure for estimating expected revenues, costs, and profitability based on usage levels and pricing strategies. As the product gains traction, data from actual sales and usage can refine these equations to create more precise projections

# Conclusion

In an era where technology is reshaping every aspect of our lives, the MediScan AI app aims to transform the way we approach healthcare. By seamlessly integrating advanced machine learning with medical expertise, we're not just providing a tool for diagnosing diseases—we're offering a smarter, more accessible way for users to manage their health.

Our app stands out by providing an innovative, cost-effective, and user-friendly alternative to traditional diagnostic methods. It empowers individuals to take charge of their health by allowing them to upload medical images and receive insights at their fingertips. Not only does this service support proactive health management, but it also promotes collaboration between patients and healthcare providers, making consultations more efficient and informed.

With a comprehensive system that encompasses a user-friendly interface, reliable logistics for report delivery, and a robust backend powered by cutting-edge CNN 21algorithms, we ensure that users have a smooth and satisfying experience. Our potential for revenue generation through premium features, consultations, and strategic partnerships is promising, providing a sustainable model for the future.

With a passionate and dedicated team at the helm, committed to development, operations, and customer relations, MediScan AI is poised to meet the evolving needs of today's health-conscious individuals. Together, we can foster a culture of informed health management, paving the way for a healthier future.