Automated intake process – Department Prediction

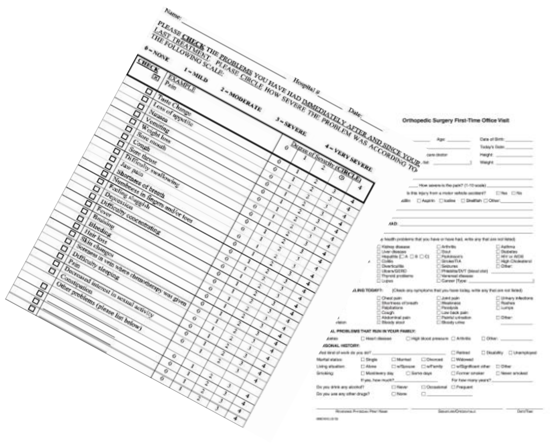
Group 10

Table of Contents

1 Introduction

As per some studies, Patients wait an hour (or more in some cases) just to be triaged, assessed, and get assigned a department based on the form they fill by the front desk of any hospital. The triage time is high because they follow a traditional process which has been handled the same way for a long time now. The traditional process is as follows:

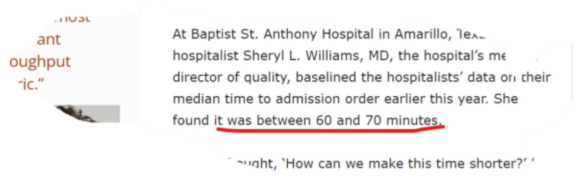
- Patient enters the hospital
- Goes to front desk and fill in an average 4-5 forms which contains approx. 100-130 Question. (This process on an average takes about 20 – 25 minutes out of the 60-70 mins of total intake allocation time)
- The forms are taken back by the triage staff and allocation (department/prognosis/specialist) is provided to the patient


Name: Hospital # Date:

PLEASE CHECK THE PROBLEMS YOU HAVE HAD IMMEDIATELY AFTER AND SINCE YOUR LAST TREATMENT. PLEASE CIRCLE HOW SEVERE THE PROBLEM WAS ACCORDING TO THE FOLLOWING SCALE:

0 = NONE 1 = MILD 2 = MODERATE 3 = SEVERE 4 = VERY SEVERE

Orthopedic Surgery First-Time Office Visit


At Baptist St. Anthony Hospital in Amarillo, Tex hospitalist Sheryl L. Williams, MD, the hospital's me director of quality, baselined the hospitalists' data on their median time to admission order earlier this year. She found it was between 60 and 70 minutes.

ught, 'How can we make this time shorter?' '

Available Solution in market: Automated solutions are already available in market which works same as traditional model. (i.e., they ask 100-120 Question on computer which in turn takes 15-20 minutes or sometimes even more) also it incurs effort which is the last things any patient wants. Moreover, these solutions ask very disease specific questions which the patient is sometimes not in a state to answer (or sometimes even do not know).

Our Solution: To design a quick, smart, and generic model which just intakes 20-25 answers (Yes/No type) and provide a department allocation to the patient by using a Machine Learning Model, with utmost accuracy and ease. For this we need to access multiple Machine learning algorithms and pick out 20-25 most weighted features. Out of these many algorithms tune each one of them for accuracy and find out the most generic features (which the patient can relate to).

Once the features are found, use these as a feed into a **fully connected neural network (FCC)** to find out the best prediction for the Department that patient need to be moved to.

On a side we also try to provide some easy dashboards to the hospital so that the hospital can manage its resources as per the load and seasons. Few simple graphs predicting seasonal change in load for diseases (trend lines) to make things ready for the load to arrive.

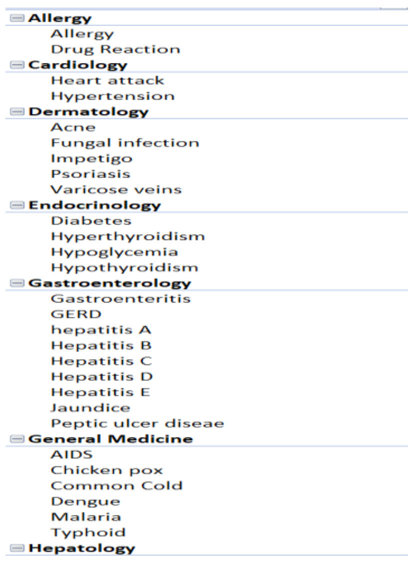
2 What’s in the Data?

Data is taken for a hospital located in the USA from Kaggle data. Dataset contains ~**130** different features that result in the occurrence of a prognosis/disease. The dataset contains a total of **40** diseases with each having a good number of samples (100-120 each). Total row count is around **5K,** and the data is spread across a period of 15 months to be exact. The data is for the year 2021 and is spread evenly for each disease.

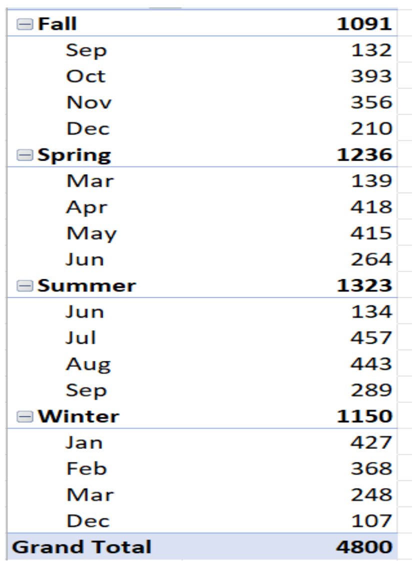
Data engineering:

Creating new columns and combining some of the classifications for making things easy to predict was the focus in this part of work

- Dates were fixed for a certain period – **Year 2021 (Jan 1st to Dec 31st)**
- 4 **Seasons** were added next to the dates.
- **Department** was added against each prognosis/disease. Research was done for each prognosis and a department was assigned. This helped to combine 40 different prognoses into 13 departments.



- **Allergy**
  - Allergy
  - Drug Reaction
- **Cardiology**
  - Heart attack
  - Hypertension
- **Dermatology**
  - Acne
  - Fungal infection
  - Impetigo
  - Psoriasis
  - Varicose veins
- **Endocrinology**
  - Diabetes
  - Hyperthyroidism
  - Hypoglycemia
  - Hypothyroidism
- **Gastroenterology**
  - Gastroenteritis
  - GERD
  - hepatitis A
  - Hepatitis B
  - Hepatitis C
  - Hepatitis D
  - Hepatitis E
  - Jaundice
  - Peptic ulcer diseae
- **General Medicine**
  - AIDS
  - Chicken pox
  - Common Cold
  - Dengue
  - Malaria
  - Typhoid
- **Hepatology**

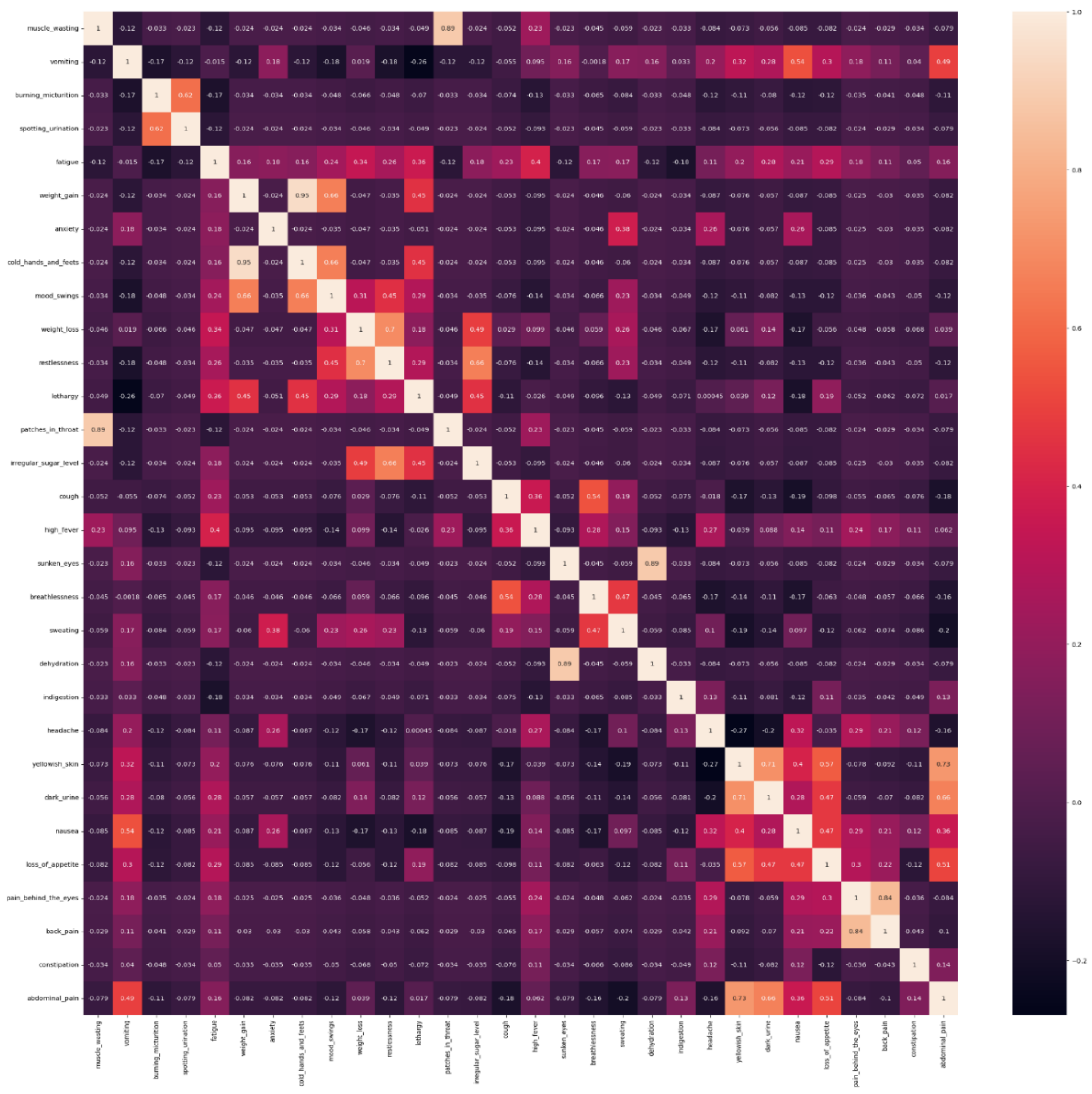


Fall	**1091**
Sep	132
Oct	393
Nov	356
Dec	210
Spring	**1236**
Mar	139
Apr	418
May	415
Jun	264
Summer	**1323**
Jun	134
Jul	457
Aug	443
Sep	289
Winter	**1150**
Jan	427
Feb	368
Mar	248
Dec	107
Grand Total	**4800**

Data cleaning:

Data cleaning is a process which is used to clean the data into usable and understandable vision. Only pertinent information is kept from the cleaned data, which may then be used to feed machine learning models for additional processing. The first step in data cleaning was to count the null values and missing values for each column in our dataset. according to the results of applying the method. When we used *‘isna (). sum ()’*, we discovered that no column contained any null values. There were no major anomalies found in the preprocessed dataset, and everything appeared to be in order. This indicates that our dataset is ready to be sent into the machine learning models for additional processing.

Co-Relation Matrix between each symptom in dataset:



According to the above matrix, there is a positive correlation between abdominal pain and yellow skin, with a value of 0.73, while anxiety has a moderate correlation with headache and sweating, with a value of 0.40, compared to other symptoms. Additionally, there is a negative correlation between headache and weight loss of around (-0.17).

3 Machine Learning Models

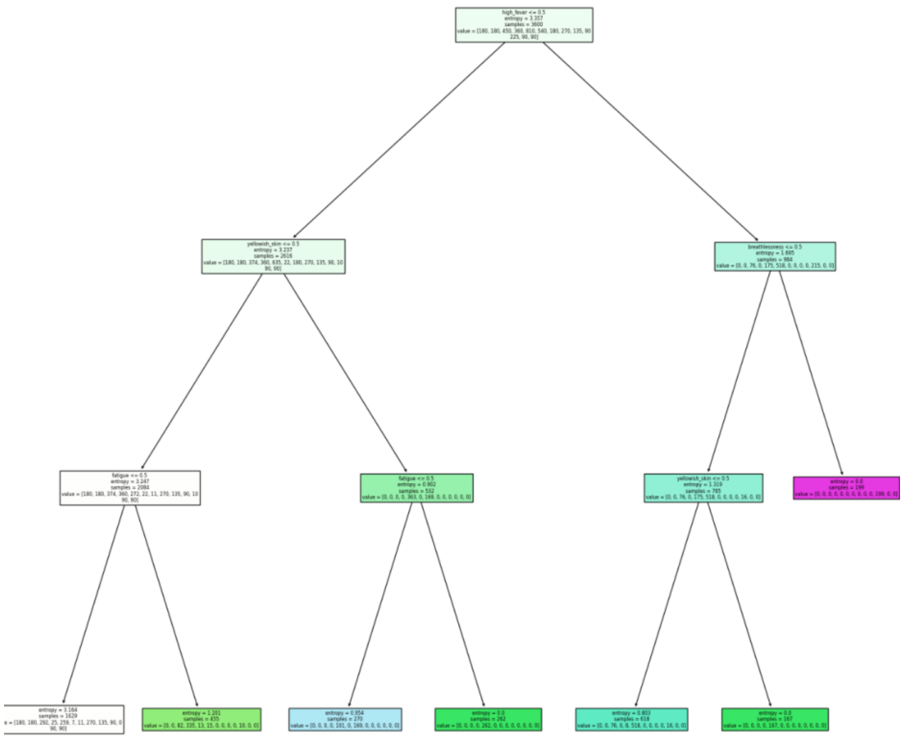
The study of algorithms that get better over time is called machine learning. Building a model that can process sample data, sometimes referred to as train data, is utilized in the artificial intelligence area to train the model to make more precise predictions for test data in the future. For this dataset, we utilized 5 machine learning models. The models are,

3.1 Decision Tree.

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving ***regression and classification problems*** too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by ***learning simple decision rules*** inferred from prior data (training data).

In Decision Trees, for predicting a class label for a record we start from the **root** of the tree. We compare the values of the root attribute with the record's attribute. Based on comparison, we follow the branch corresponding to that value and jump to the next node.

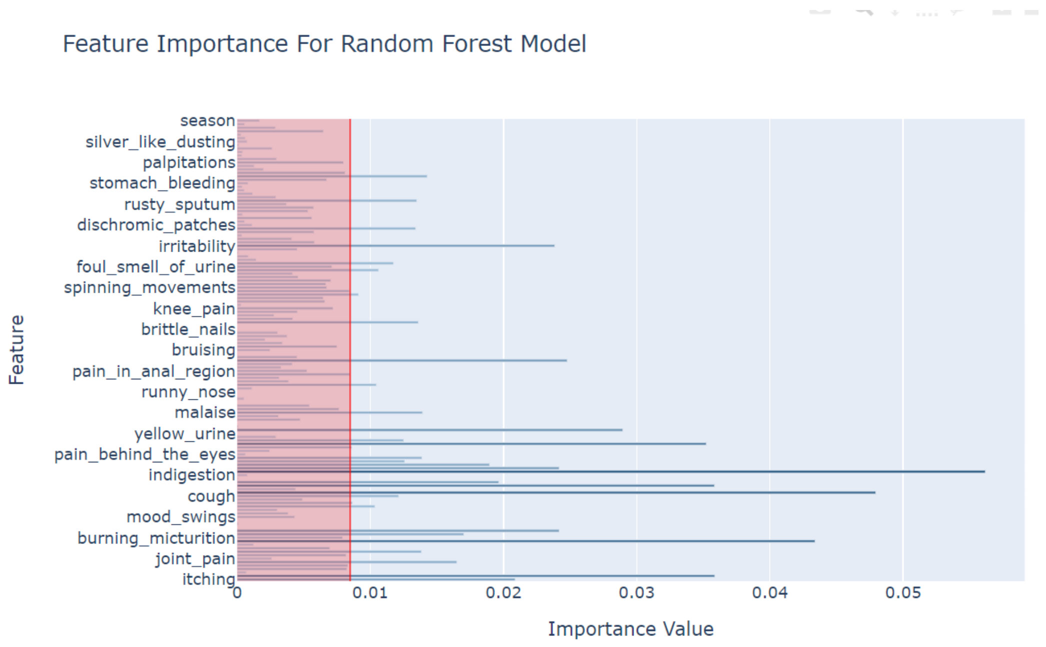


high_fever <= 0.5
entropy = 3.357
samples = 3600
value = [180, 180, 450, 360, 810, 540, 180, 270, 135, 90
225, 90, 90]
yellowish_skin <= 0.5
entropy = 3.237
samples = 2616
value = [180, 180, 374, 360, 635, 22, 180, 270, 135, 90, 10
90, 90]
breathlessness <= 0.5
entropy = 1.695
samples = 984
value = [0, 0, 76, 0, 175, 518, 0, 0, 0, 0, 215, 0, 0]
fatigue <= 0.5
entropy = 3.247
samples = 2084
value = [180, 180, 374, 360, 272, 22, 11, 270, 135, 90, 10
90, 90]
fatigue <= 0.5
entropy = 0.902
samples = 532
value = [0, 0, 0, 0, 363, 0, 169, 0, 0, 0, 0, 0, 0]
yellowish_skin <= 0.5
entropy = 1.319
samples = 785
value = [0, 0, 76, 0, 175, 518, 0, 0, 0, 0, 16, 0, 0]
entropy = 0.0
samples = 199
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 199, 0, 0]
entropy = 3.164
samples = 1629
e = [180, 180, 292, 25, 259, 7, 11, 270, 135, 90, 0
90, 90]
entropy = 1.201
samples = 455
value = [0, 0, 82, 335, 13, 15, 0, 0, 0, 0, 10, 0, 0]
entropy = 0.954
samples = 270
value = [0, 0, 0, 0, 101, 0, 169, 0, 0, 0, 0, 0, 0]
entropy = 0.0
samples = 262
value = [0, 0, 0, 0, 262, 0, 0, 0, 0, 0, 0, 0, 0]
entropy = 0.803
samples = 618
value = [0, 0, 76, 0, 8, 518, 0, 0, 0, 0, 16, 0, 0]
entropy = 0.0
samples = 167
value = [0, 0, 0, 0, 167, 0, 0, 0, 0, 0, 0, 0, 0]

3.2 Random forest:

Random forest is a ***Supervised Machine Learning Algorithm*** that is ***used widely in Classification and Regression problems***. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most notable features of the Random Forest Algorithm is that it can handle the data set containing ***continuous variables*** as in the case of regression and ***categorical variables*** as in the case of classification. It performs better results for classification problems.

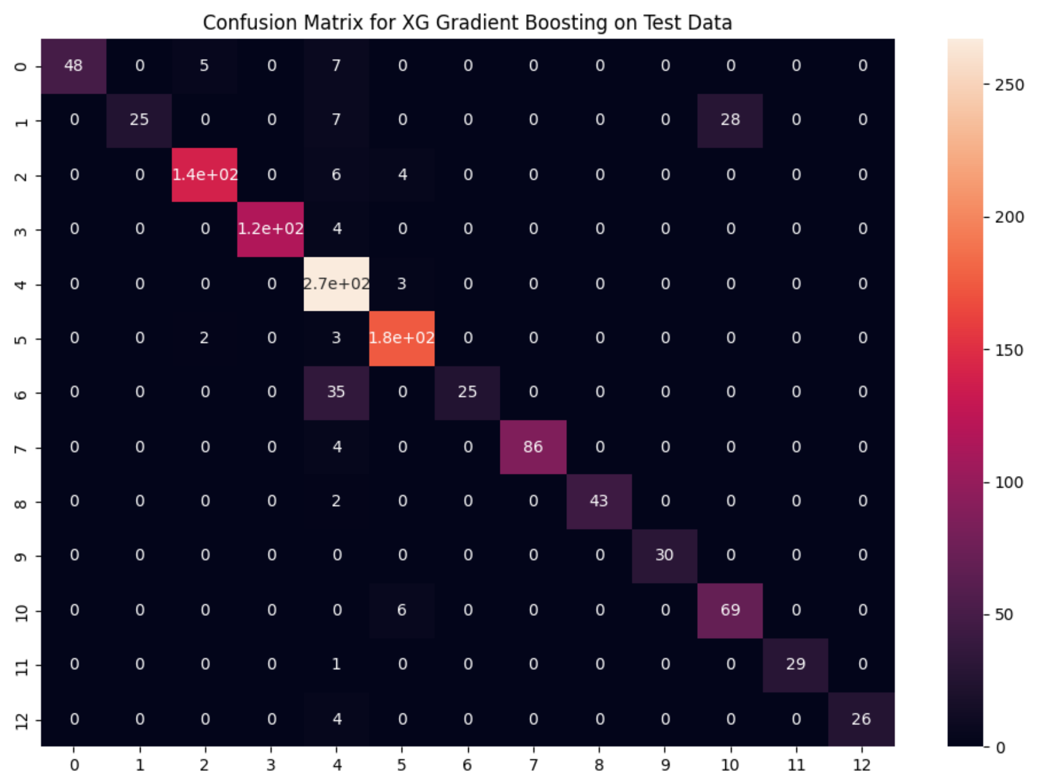


3.3 XG Gradient boosting:

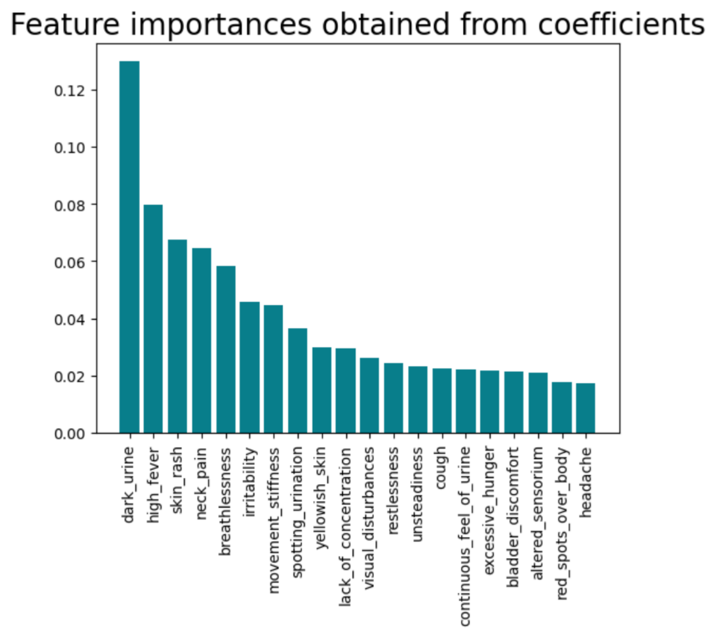
Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model to minimize the error. How are the targets calculated? The target outcome for each case in the data depends on how much changing that case's prediction impacts the overall prediction error:

- If a slight change in the prediction for a case causes a large drop in error, then next target outcome of the case is a high value. Predictions from the new model that are close to its targets will reduce the error.
- If a minor change in the prediction for a case causes no change in error, then next target outcome of the case is zero. Changing this prediction does not decrease the error.

The name *gradient boosting* arises because target outcomes for each case are set based on the gradient of the error with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of predictions for each training case.



Many of the departments in this case are accurately predicated for test data, except for three departments with the symptoms of the urology department being predicted in the general medicine department due to an imbalance of data.



3.4 Logistic Regression

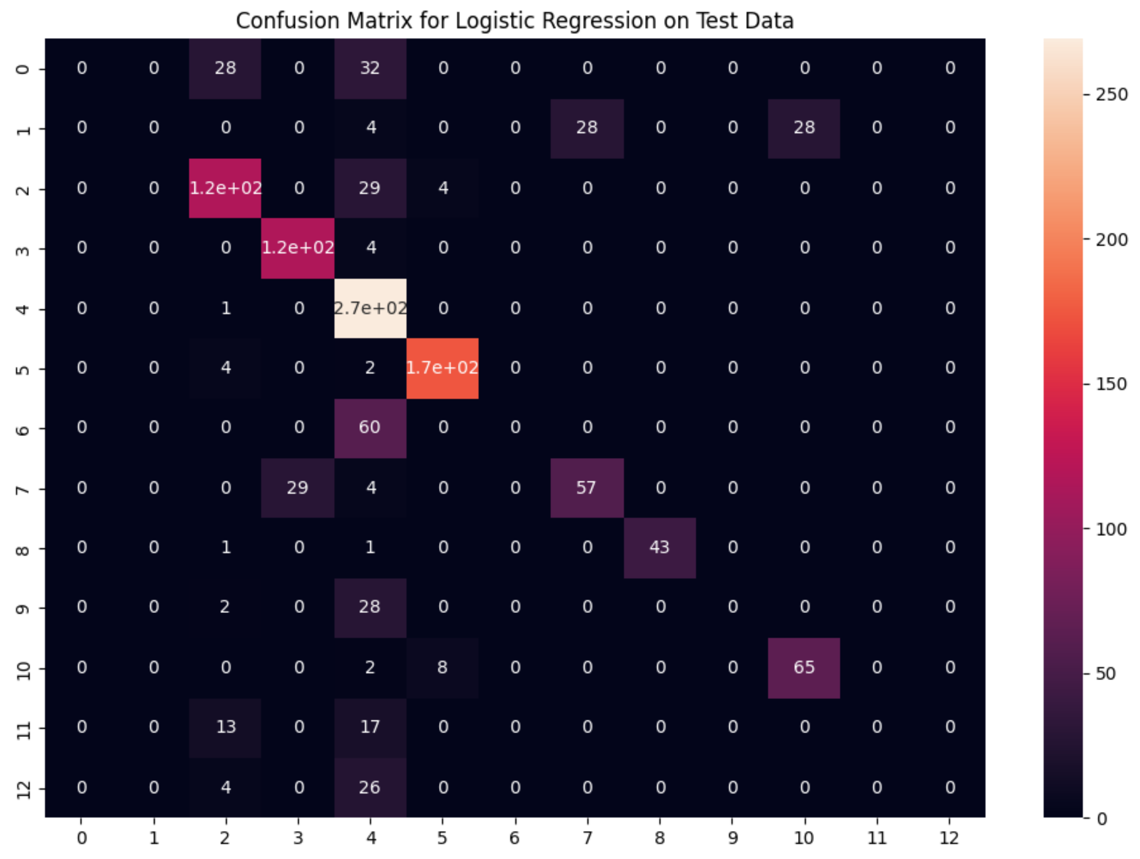
Logistic regression is a statistical model that Is used to determine the probability that an event will happen. It shows the relationship between features, and then calculates the probability of a certain outcome.

Logistic regression is used in ML to help create accurate predictions. It is like linear regression, except rather than a graphical outcome, the target variable is binary; the value is either one, or zero.

There are two types of measurables, the explanatory variables/ features (item being measured) and the response variable/ target binary variable, which is the outcome.

For example, when trying to predict whether a student will pass or fail a test, the hours studied are the feature, and the response variable will have two values - pass or fail.

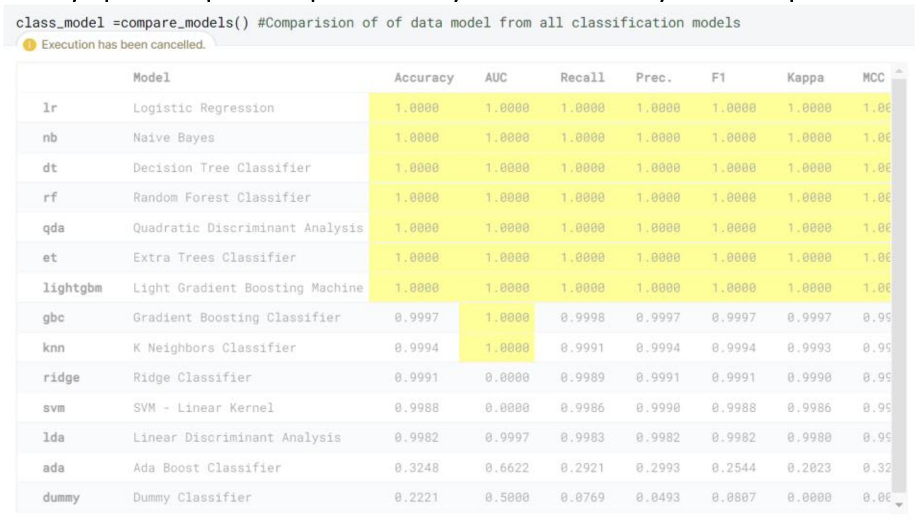
Confusion Matrix:



Many departments are incorrectly predicted using logistic regression on test data, particularly the general medicine department because each department's model classified a sample, giving patients inaccurate results and inaccurate information, while the endocrinology department is correctly predicted

3.5 Pycaret Analysis

PyCaret is an open-source, low-code machine learning library in Python that automates machine learning workflows. It is an end-to-end machine learning and model management tool that exponentially speeds up the experiment cycle and makes you more productive.



```
class_model =compare_models() #Comparision of of data model from all classification models
```

Execution has been cancelled.

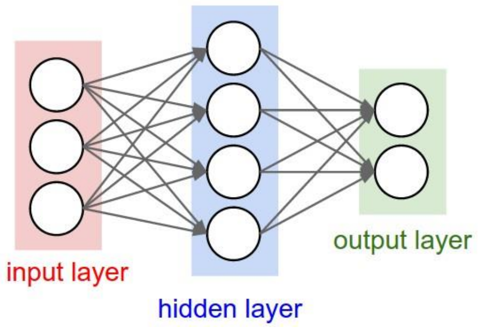
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lr	Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.00
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.00
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.00
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.00
qda	Quadratic Discriminant Analysis	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.00
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.00
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.00
gbc	Gradient Boosting Classifier	0.9997	1.0000	0.9998	0.9997	0.9997	0.9997	0.99
knn	K Neighbors Classifier	0.9994	1.0000	0.9991	0.9994	0.9994	0.9993	0.99
ridge	Ridge Classifier	0.9991	0.0000	0.9989	0.9991	0.9991	0.9990	0.99
svm	SVM - Linear Kernel	0.9988	0.0000	0.9986	0.9990	0.9988	0.9986	0.99
lda	Linear Discriminant Analysis	0.9982	0.9997	0.9983	0.9982	0.9982	0.9980	0.99
ada	Ada Boost Classifier	0.3248	0.6622	0.2921	0.2993	0.2544	0.2023	0.32
dummy	Dummy Classifier	0.2221	0.5000	0.0769	0.0493	0.0807	0.0000	0.00

Pycaret analysis came in handy to identify the top classification algorithms, also it helped in carrying out the top 4 approach for the methods. The accuracy with Pycaret algorithms were high than the individual tried algorithms above but still there were some shortcomings which led to dropping the approach

1. Blackbox approach – lack of opportunity to tune hyperparameters.
2. Feature Encoding – forced feature encoding and using more features just to achieve accuracy was something team
3. Accuracy as focus - Most of the algorithms concentrated on accuracy rather than feature weightage and ease of use (which being two big focus points of the project)

3.6 **Fully Connected Deep learning**:

As part of Step 2 (after finding the best ML algorithm and identifying 20 best features), a fully connected model is deployed with an intention to use all the 20 features equally and classify the department as per the 20 identified features



4 Results

After passing the data from all the various steps, below results were achieved, at first level a comparison between all the used ML algorithms is made to find out the best accuracy and Feature weightage as per involvement was identified

4.1 Comparing accuracies of all the mode

Model	Accuracy	Recall	Precision	F1	No.of features
Logistic Regression	70.1%	70.1%	70.1%	70.1%	55/133
Decision Tree	52.7%	52.7%	52.7%	52.7%	133/133
Random Forest	100.0%	100.0%	100.0%	100.0%	133/133
XG Gradient Boosting	89.9%	89.9%	89.9%	89.9%	133/133

Interpretation: From the above chart, we can see the comparison of precision, recall, accuracy and F1 scores Of XGBoost and Random Forest. We can see that the scores are similar.

4.2 Best 20 Features

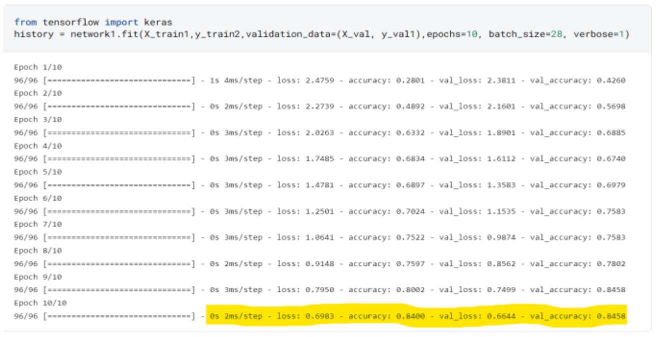
After identifying the two algorithms, features were examined and as per relevance below 20 features were identified to be part of the final classification algorithm

- Headache
- Cough
- High fever
- breathlessness
- unsteadiness
- skin_rash
- red_spots_over_body
- yellowish_skin
- bladder_discomfort
- continuous_feel_of_urine
- dark_urine
- spotting_urination
- excessive_hunger
- neck_pain
- movement_stiffness
- irritability
- lack_of_concentration
- visual_disturbances
- Restlessness
- altered_sensorium

4.3 Deep Learning Work

Now the set of chosen 20 features are fed into fully connected deep learning network, with a '**relu'** function to predict the accuracy of the outcome on the validation data. Tests were run on test set as well as live inputs.

An overall accuracy of **85%** was achieved on the same

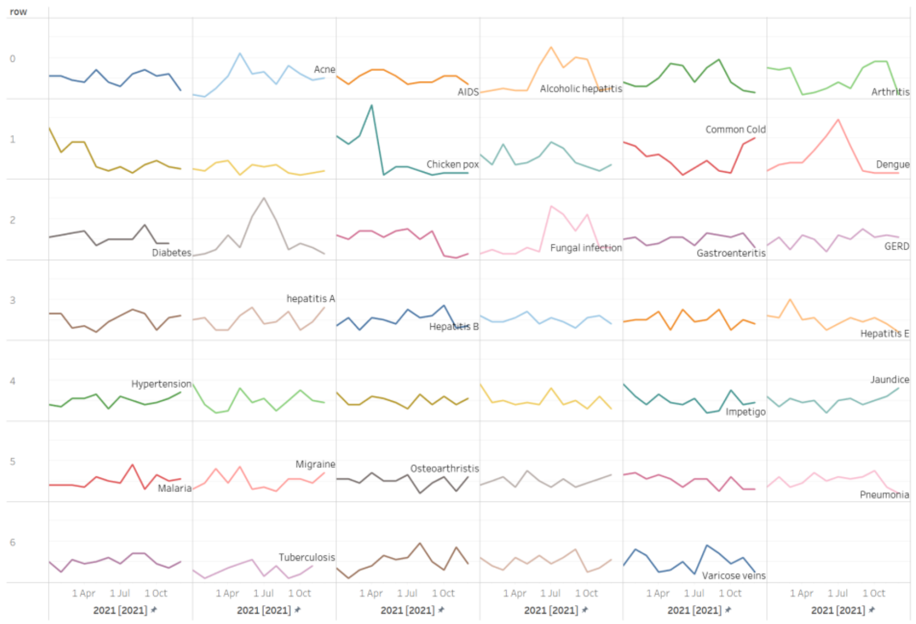


```
from tensorflow import keras
history = network1.fit(X_train1,y_train2,validation_data=(X_val, y_val1),epochs=10, batch_size=28, verbose=1)
```

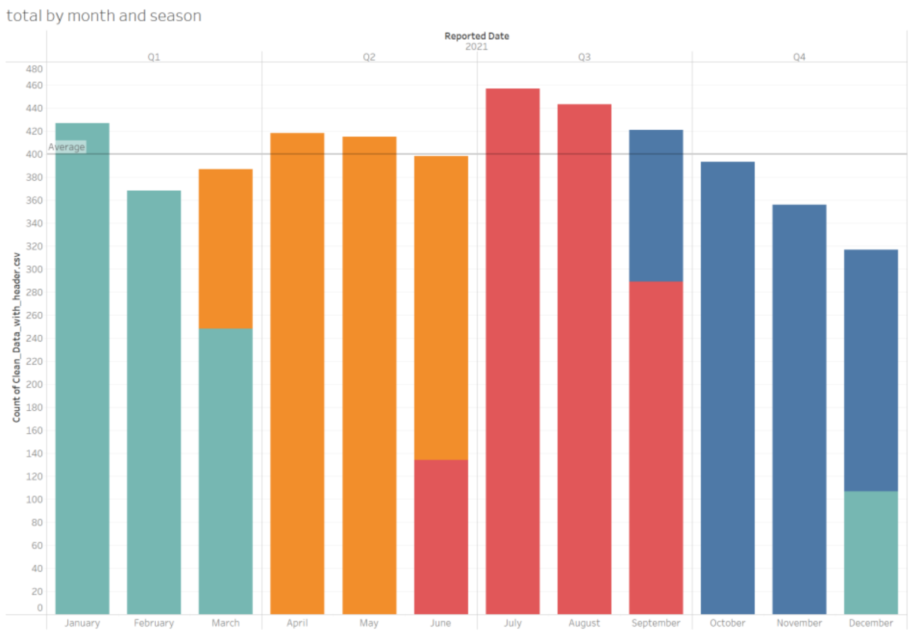
```
Epoch 1/10
96/96 [==============================] - 1s 4ms/step - loss: 2.4759 - accuracy: 0.2801 - val_loss: 2.3811 - val_accuracy: 0.4260
Epoch 2/10
96/96 [==============================] - 0s 2ms/step - loss: 2.2739 - accuracy: 0.4892 - val_loss: 2.1601 - val_accuracy: 0.5698
Epoch 3/10
96/96 [==============================] - 0s 3ms/step - loss: 2.0263 - accuracy: 0.6332 - val_loss: 1.8901 - val_accuracy: 0.6885
Epoch 4/10
96/96 [==============================] - 0s 3ms/step - loss: 1.7485 - accuracy: 0.6834 - val_loss: 1.6112 - val_accuracy: 0.6740
Epoch 5/10
96/96 [==============================] - 0s 3ms/step - loss: 1.4781 - accuracy: 0.6897 - val_loss: 1.3583 - val_accuracy: 0.6979
Epoch 6/10
96/96 [==============================] - 0s 3ms/step - loss: 1.2501 - accuracy: 0.7024 - val_loss: 1.1535 - val_accuracy: 0.7583
Epoch 7/10
96/96 [==============================] - 0s 3ms/step - loss: 1.0641 - accuracy: 0.7522 - val_loss: 0.9874 - val_accuracy: 0.7583
Epoch 8/10
96/96 [==============================] - 0s 2ms/step - loss: 0.9148 - accuracy: 0.7597 - val_loss: 0.8562 - val_accuracy: 0.7802
Epoch 9/10
96/96 [==============================] - 0s 3ms/step - loss: 0.7950 - accuracy: 0.8002 - val_loss: 0.7499 - val_accuracy: 0.8458
Epoch 10/10
96/96 [==============================] - 0s 2ms/step - loss: 0.6983 - accuracy: 0.8400 - val_loss: 0.6644 - val_accuracy: 0.8458
```

4.4 Tableau Work

Some Tableau graphs were also created from the final set of data for the Hospital to get some trends and breakdowns of the diseases in the full year (2021), some of the below graphs are explained

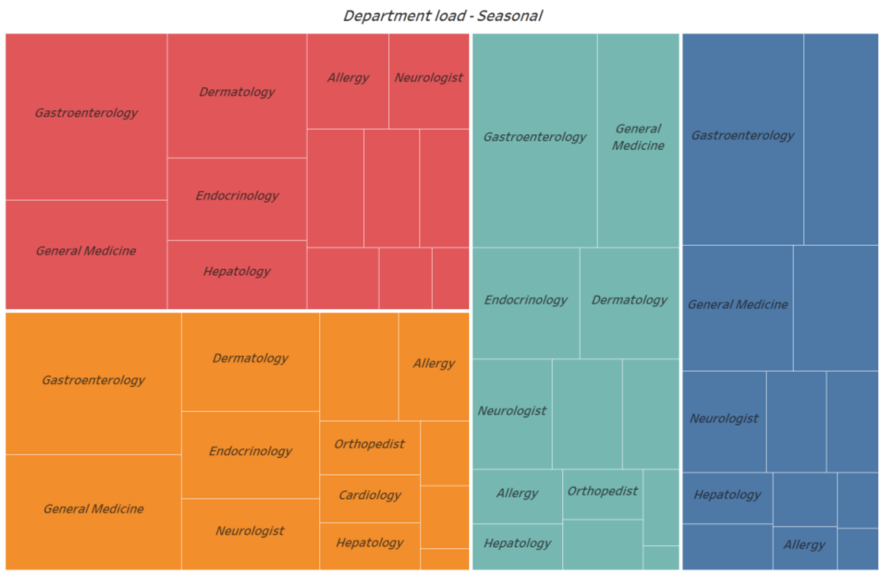


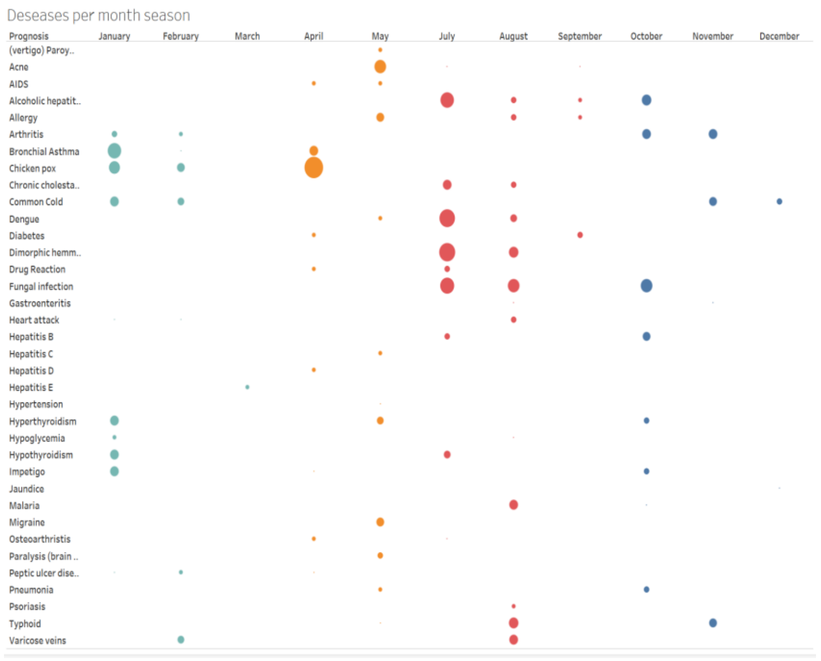
A Trellis chart for each Prognosis and its yearly Trends. This type of chart can help identify the spikes in each month and for each prognosis. Moving specialists for certain month and helping load management monthly



Overall load of patients in every month/season. Showing against the total average helps to predict the busiest month and season

Seasonal breakdown of the Departments in a Tree map graph. This shows the busiest department in each of the four seasons. This can help Hospital to move resources between departments in every season





Another way of showing the overall patient load but this time on each prognosis level, filters are provided which can set baselines. Also, easy presentation with size bubbles makes visualization easy on the eyes.

5 Challenges and Solutions

Over the last 2 months team went through multiple challenges and were ably planned and worked around the problems to achieve the desired goal

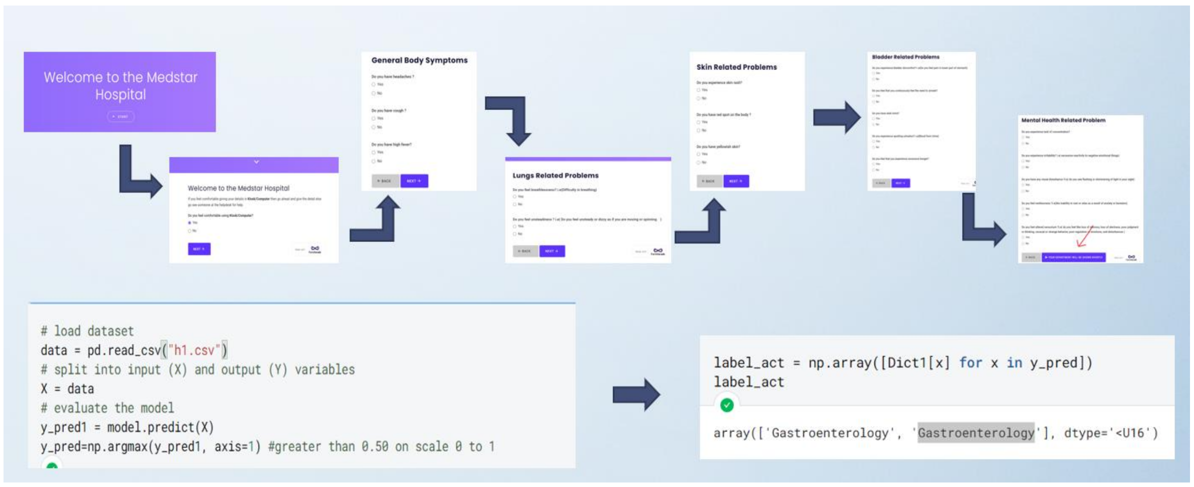
- **Exploring too many algorithms – ML and resources:** team went through 6-7 algorithms to get the right set of results, tuning parameters and comparing results added a lot of initial time to the effort. After a while, the team settled for top 4 to cut the time and concentrate on a set to find solutions
- **PyCaret and analysis – breaking the Blackbox:** team added Pycaret to find a consolidated solution and check if we were moving in the right direction, but autoML

has its own challenges of encoding, no parameter changing/tuning and pulling algorithms where team was not comfortable to work on. Team worked on Pycaret for a week and once challenges surfaced moved to manual approach

- **Data engineering work – Defining the rules:** defining the right variables like Season and department took some time as one of the focuses was to reduce the classifications and deciding between month and season took some time.
- **Lack of healthcare knowledge – depending on Google all the time:** team being not from healthcare background had to depend on online search and google for deciding crucial factors like 'Department'
- **Data Imbalance:** Initial dataset had 40 diseases evenly spread over 4800 rows (120 each), this created a bit of a bias and makes the algorithm not learn on the variance. As a solution team categorized prognosis into departments which removed this bias to some extent as now the spread had variance, this problem took a lot of time to solve and some advice from the Project mentor helped in breaking the solution. The solution of adding department also helped in increasing the accuracy of the results on validation

6 Final Work

A demo dashboard (with Google forms) is created which act as an intake form for the patient (just 20 questions), this form acts as input for the level 2 algorithm to predict the 'department', the patient needs to be moved to

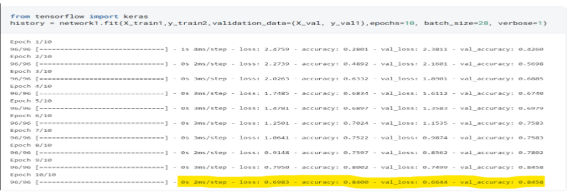


```
# load dataset
data = pd.read_csv("h1.csv")
# split into input (X) and output (Y) variables
X = data
# evaluate the model
y_pred1 = model.predict(X)
y_pred=np.argmax(y_pred1, axis=1) #greater than 0.50 on scale 0 to 1
```

```
label_act = np.array([Dict1[x] for x in y_pred])
label_act
```

```
array(['Gastroenterology', 'Gastroenterology'], dtype='<U16')
```

Model	Accuracy	Recall	Precision	F1	No.of features
Logistic Regression	70.1%	70.1%	70.1%	70.1%	55/133
Decision Tree	52.7%	52.7%	52.7%	52.7%	133/133
Random Forest	100.0%	100.0%	100.0%	100.0%	133/133
XG Gradient Boosting	89.9%	89.9%	89.9%	89.9%	133/133
Pycaret Models	100.0%	100.0%	100.0%	100.0%	133/133
Fully Connected Model (relu)	87.2%	87.2%	87.2%	87.2%	133/133



```
from tensorflow import keras
history = network1.fit(X_train1,y_train2,validation_data=(X_val, y_val1),epochs=10, batch_size=28, verbose=1)
```

```
Epoch 1/10
96/96 [==============================] - 1s 4ms/step - loss: 2.4759 - accuracy: 0.2801 - val_loss: 2.3811 - val_accuracy: 0.4260
Epoch 2/10
96/96 [==============================] - 0s 2ms/step - loss: 2.2739 - accuracy: 0.4892 - val_loss: 2.1601 - val_accuracy: 0.5698
Epoch 3/10
96/96 [==============================] - 0s 3ms/step - loss: 2.0263 - accuracy: 0.6332 - val_loss: 1.8901 - val_accuracy: 0.6885
Epoch 4/10
96/96 [==============================] - 0s 3ms/step - loss: 1.7485 - accuracy: 0.6834 - val_loss: 1.6112 - val_accuracy: 0.6740
Epoch 5/10
96/96 [==============================] - 0s 3ms/step - loss: 1.4781 - accuracy: 0.6897 - val_loss: 1.3583 - val_accuracy: 0.6979
Epoch 6/10
96/96 [==============================] - 0s 3ms/step - loss: 1.2501 - accuracy: 0.7024 - val_loss: 1.1535 - val_accuracy: 0.7583
Epoch 7/10
96/96 [==============================] - 0s 3ms/step - loss: 1.0641 - accuracy: 0.7522 - val_loss: 0.9874 - val_accuracy: 0.7583
Epoch 8/10
96/96 [==============================] - 0s 2ms/step - loss: 0.9148 - accuracy: 0.7597 - val_loss: 0.8562 - val_accuracy: 0.7802
Epoch 9/10
96/96 [==============================] - 0s 3ms/step - loss: 0.7950 - accuracy: 0.8002 - val_loss: 0.7499 - val_accuracy: 0.8458
Epoch 10/10
96/96 [==============================] - 0s 2ms/step - loss: 0.6983 - accuracy: 0.8400 - val_loss: 0.6644 - val_accuracy: 0.8458
```

7 Team's individual Contributions

Name	Contribution
Amit Sharma - 0794488	Project: • Working around Data – EDA • Data engineering and creating SQL queries for the data manipulations • Initial Algorithm selection and Pycaret code for the multiple algorithms. • Working on one of the monthly prognosis load Tableau graphs Presentation: • Creating the presentation and the theme Report: • Collating the numbers and creating the report work
Rajan Atulkumar Patel - 0789953	Project: • Working on the first deep learning model (with 133 features) • Helped in ML algorithms and EDA of the initial data Survey • Created the survey which can act as the demo for the front end (kiosk) • Helped Darshan with Tableau Graphs Report: • Adding details for the skeleton of the report • Adding codebase work and documentation
Darshan Patel - 0789282	Project • Participated in ideation and algorithm training • Participated in Date handling of the EDA process Tableau • Created majority of the Tableau Graphs from the raw data, Trellis, Bars and Treemap for the seasonal and prognosis data Report: • Added Graphs/screens and added code in Git for team's usage
Urali Virajsinh Rana- 0789945	Project: • Project technical handler with expertise in python • Participated in getting data in shape and creating correlational matrices • Algorithm and Feature selection methodology Presentation: • Added numbers and stats in the presentations • Challenge definitions and solutions for the same Report: • Algorithm details • Confusion Matrix • Feature selection

8 References

- **Dataset (Kaggle):**
  https://www.kaggle.com/datasets/neelima98/disease-prediction-using-machine-learning
- **Prognosis:**
  https://medical-dictionary.thefreedictionary.com/prognosis
- **Tableau**
  https://www.tableau.com/learn
- **Machine learning:**
  - https://keras.io/api/
  - https://numpy.org/doc/
  - Grokking Deep Learning, by Andrew Trask
  - Deep Learning with Python, Second Edition, by Francois Chollet
- **Images:**
  Images by Unknown Author is licensed under CC BY (Microsoft PowerPoint)

9 Appendices

- **Group10FinalReport.docx:** Contains the final report
- **DeepNote:** Link
- **GitHub Repo: https://github.com/miners-work/HealthCare**
  - Repo contains all documents, codes, tableau files, status reports, presentation, and datasets
- **Demo Front end:**
  https://formfacade.com/public/112670540446610566057/all/form/1FAIpQLSc2nwZx33FZa9NFha4ra58Zdm9HlBnAvHcUmx2jHsUN1cr6yg