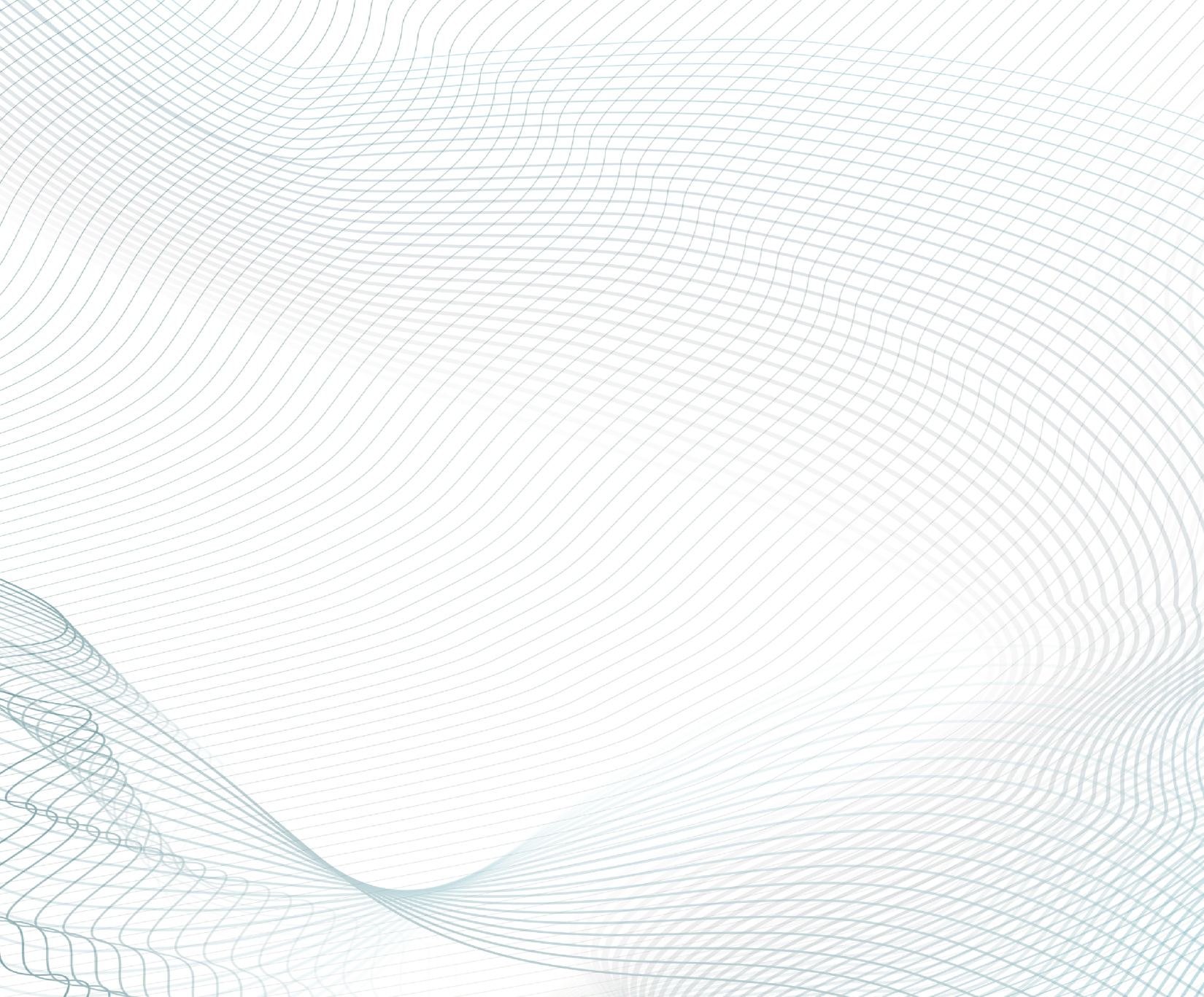


|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | FINAL PROJECT REPORT | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | CRITICALITY INDEX PREDICTION USING MACHINE LEARNING |  | | | |
|  | | | | **Date****22/15/2022****Professor’s Name** **Mahmoud Artima** |  | | | |
|  | | |  | | |  | | |



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  | | |  |
|  | Introduction | | | | | | |  |
|  |  | | |  |  | | |  |
|  |  | |  | | |  | |  |
|  |  |  | In this project, our main motive is to predict the criticality of a patient using Machine Learning modules.  This model supports the doctors with the best decision-making in any given situation. For this model building, we are using the data from the Mimic Clinical dataset. We have applied for and got access to use the MIMIC-III Clinical database which is very important as it contains sensitive data.  We used Big Query and GCP to download and load the data for analysis and performed numerous analytics on the MIMIC III database to understand the relations, dependencies, correlations, and measurements. | | |  |  |  |
|  | | | | |
|  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | Related Work | | |  | |
|  | |  |  |  |  | |
|  | Analysis: All the entity relationships and database details in this project are taken from the Mimic-iii schema and Entity Relationship on the below page.  <https://mit-lcp.github.io/mimic-schema-spy/index.html>  Also, the dataset can be found using the below URL.  <https://physionet.org/content/mimiciii/1.4/>  We do not have any column specifically for Criticality Index in the MIMIC dataset. So, we built a Criticality Index variable using the most specific and important factors which can predict the criticality most accurately.  Let’s see the step-by-step procedure followed to get the required data.  We have many limitations to using Big Query without a Billing account, so we used only Select commands and Temporary Tables to acquire data.  **Table 1: Standardized Vitals Table**  We have selected the top 6 vitals which play a crucial role in assessing a patient’s health and filtered those vitals along with readings for each patient from chartevents & d\_items tables.  Also standardized the measurements for Temperature & Oxygen Saturation for easy calculations and saved the results to temporary table **“Vitals\_step1”**   * Converted all the Temperatures in Fahrenheit to Celsius. * Converted all the Oxygen Saturation in % to a decimal value.   **Query:**  Select ce.icustay\_id -- icu stay id, ce.HADM\_ID,  ce.charttime -- date/time when reading was taken  ,di.label -- vital name  ,ce.value as vital\_reading -- vital reading  ,di.CATEGORY as High\_Level\_Vital\_Name  ,di.UNITNAME, case  when UNITNAME = 'F' and SAFE\_CAST(REGEXP\_REPLACE(value,'[^0-9.]','') AS FLOAT64) > 79 then (SAFE\_CAST(REGEXP\_REPLACE(value,'[^0-9.]','') AS FLOAT64) - 32) \* 5./9  when label in ('Arterial O2 Saturation', 'O2 saturation pulseoxymetry', 'SpO2','SaO2') and SAFE\_CAST(REGEXP\_REPLACE(value,'[^0-9.]','') AS FLOAT64) <= 1 then SAFE\_CAST(REGEXP\_REPLACE(value,'[^0-9.]','') AS FLOAT64) \* 100  else SAFE\_CAST(REGEXP\_REPLACE(value,'[^0-9.]','') AS FLOAT64) end  from `physionet-data.mimiciii\_clinical.chartevents` ce  join `physionet-data.mimiciii\_clinical.d\_items` di ON ce.itemid = di.itemid  where di.LABEL in ('Arterial BP [Systolic]','Manual BP [Systolic]', 'Arterial BP #2 [Systolic]', 'Manual Blood Pressure Systolic Left','Manual Blood Pressure Systolic Right', 'Manual Blood Pressure Diastolic Right', 'Manual Blood Pressure Diastolic Left', 'Arterial Blood Pressure systolic', 'Arterial Blood Pressure diastolic', 'ART BP Systolic', 'ART BP Diastolic', 'ABP [Diastolic]', 'Arterial BP [Diastolic]', 'Manual BP [Diastolic]', 'Arterial BP #2 [Diastolic]', 'Heart Rate', 'Spont RR', 'Resp Rate (Total)', 'Respiratory Rate', 'Respiratory Rate (Total)', 'PULSE', 'Temperature Celsius', 'Temperature Fahrenheit', 'Temperature F', 'Temperature C', 'Temperature C (calc)', 'Temperature F (calc)', 'Oxygen Saturation', 'SpO2', 'SaO2', 'Arterial O2 Saturation','O2 saturation pulseoxymetry', 'Breath Rate');  **Table 2: Joining the Step1 Temp, Admissions & callout tables**  Based on the vital readings and using the HADM\_ID as unique key we are joining the admissions & Callout tables to filter the data such as admission\_type, diagnosis, careunit, and wardid and saved the results to the temporary table **“Step2”**  **Query:**  SELECT sv.HADM\_ID, sv.icustay\_id, sv.charttime, sv.f0\_ as vital\_reading, sv.High\_Level\_Vital\_Name, sv.label, sv.UNITNAME,  ad.admission\_type, ad.diagnosis, a.submit\_careunit, a.callout\_wardid  FROM `healthcare-366817.Temporary.Vitals\_step1` sv  join `physionet-data.mimiciii\_clinical.admissions` ad  ON sv.HADM\_ID = ad.HADM\_ID  join `physionet-data.mimiciii\_clinical.callout` a  ON sv.HADM\_ID = a.HADM\_ID  **Table 3: Create an age & Length of stay:**  Using the patient id filtering the data such as Gender, and length of stay(los) and calculated the age of the patient by finding the difference between the date of join & date of birth and saved the results to the temporary table “AGE”  **Query:**  SELECT icu.HADM\_ID, pat.GENDER, icu.los,  DATE\_DIFF(DATE(icu.intime), DATE(pat.dob), YEAR) AS age  FROM `physionet-data.mimiciii\_clinical.icustays` AS icu  JOIN `physionet-data.mimiciii\_clinical.patients` AS pat  ON icu.subject\_id = pat.subject\_id  **Table 4: Joining the above 2 tables**  Joining the tables AGE & Step2 on HADM\_ID and saved the results to temporary table “Step3”  **Query:**  SELECT DISTINCT sv.HADM\_ID, sv.icustay\_id, sv.charttime, sv.vital\_reading, sv.High\_Level\_Vital\_Name, sv.label, sv.UNITNAME,  sv.admission\_type, sv.diagnosis, sv.submit\_careunit, sv.callout\_wardid,  ad.gender, ad.age, ad.los  FROM `healthcare-366817.Temporary.Step2` sv  INNER join `healthcare-366817.Temporary.AGE` ad  ON sv.HADM\_ID = ad.HADM\_ID  **Table 5: Joining Stpe3 with Labevents**  This is the final table where we are filtering the lab values, unit of measurements and flag warnings & itemid.  Also, we have limit the results to 39000000 due to limitations in BigQuery.  **Query:**  SELECT ab.HadM\_ID, ab.icustay\_id, ab.charttime, ab.vital\_reading, ab.High\_Level\_Vital\_Name, ab.label, ab.UNITNAME,  ab.admission\_type, ab.diagnosis, ab.submit\_careunit, ab.callout\_wardid,  ab.gender, ab.age, ab.los, lb.valuenum, lb.valueuom, lb.flag,  lb.itemid as labevents\_itemid  FROM `healthcare-366817.Temporary.Step3` ab  INNER join `physionet-data.mimiciii\_clinical.labevents` lb  ON ab.HadM\_ID = lb.HadM\_ID  LIMIT 39000000 Methods **Data Cleaning:**  **Tools Used: Big Query and Python (Jupyter Notebook)**  After cleaning the data and storing it in the Big Query Temporary table we exported it to GCS Bucket in CSV format, which was further stored into multiple files, i.e. almost 25 files.  Then we imported the CSV files, merged them into 1 file using Python, and stored it in the below google drive shared location for the team to access.    <https://drive.google.com/drive/folders/10cHSCimSXGBu6RyQTTPYuGYw7Y-9ccKx>  **Step-1:**  Combined all the same Vital Reading Items into a single Item.   * All the below Vital Readings are renamed to **“Systolic BP”.**   'Arterial BP [Systolic]', 'Arterial BP #2 [Systolic]', 'Arterial Blood Pressure systolic', 'ART BP Systolic', 'Manual BP [Systolic]', 'Manual Blood Pressure Systolic Left', 'Manual Blood Pressure Systolic Right'  **Unit of Measurement: mmHg**   * All the below Vital Readings are renamed to **“Diastolic BP”**.   'Arterial Blood Pressure diastolic', 'ART BP Diastolic', 'ABP [Diastolic]', 'Arterial BP [Diastolic]', 'Arterial BP #2 [Diastolic]', 'Manual Blood Pressure Diastolic Right', 'Manual Blood Pressure Diastolic Left', 'Manual BP [Diastolic]'  **Unit of Measurement: mmHg**   * All the below Vital Readings are renamed to **“Respiratory Rate”.**   'Resp Rate (Total)', 'Respiratory Rate', 'Respiratory Rate (Total)', ‘Spont RR’, ‘Breath Rate’  **Unit of Measurement: insp/min**   * All the below Vital Readings are renamed to **“Temperature C”.**   'Temperature Celsius', 'Temperature Fahrenheit', 'Temperature F', 'Temperature C’, ‘Temperature C (calc)', 'Temperature F (calc)'  **Unit of Measurement: Celsius**   * All the below Vital Readings are renamed to **“Oxygen Saturation”**   'Oxygen Saturation', 'SpO2', 'SaO2', 'Arterial O2 Saturation',  'O2 saturation pulseoxymetry'  **Unit of Measurement: Decimal numbers**   * All the Vital Readings which are PULSE & Heart Rate are renamed to **“Heart Rate”**   **Unit of Measurement:** BPM  **Step 2:**   * Dropped the Null values and updated the Vital Readings labels with single unique names. * Arterial O2 Saturation, O2 saturation pulseoxymetry, SpO2,SaO2 is in %. Converted all percentage values into decimal values * Temperatures are in a combination of Fahrenheit and Celsius. Converted all the values to Celsius * Removed the Outliers in the Age column using the below command.   proj2 = proj2.drop(proj2[proj2.age > 100].index)  **Data Engineering:**  **Step 1:**  We have converted the labels column with 6 vital labels into six unique columns for each vital and the vital readings accordingly using the below queries.  **Respiratory Rate:**  def xyz(proj2):  if (proj2["label"] == 'Respiratory Rate'):  return proj2["vital\_reading"]  else :  return "0"  proj2['Respiratory\_Rate'] = proj2.apply(lambda proj2:xyz(proj2), axis=1)  **Oxygen Saturation:**  def xyz(proj2):  if (proj2["label"] == 'Oxygen Saturation'):  return proj2["vital\_reading"]  else :  return "0"  proj2['Oxygen\_Saturation'] = proj2.apply(lambda proj2:xyz(proj2), axis=1)  **Diastolic BP:**  def xyz(proj2):  if (proj2["label"] == 'Diastolic BP'):  return proj2["vital\_reading"]  else :  return "0"  proj2['Diastolic\_BP'] = proj2.apply(lambda proj2:xyz(proj2), axis=1)  **Systolic BP:**  def xyz(proj2):  if (proj2["label"] == 'Systolic BP'):  return proj2["vital\_reading"]  else :  return "0"  proj2['Systolic\_BP'] = proj2.apply(lambda proj2:xyz(proj2), axis=1)  **Heart Rate:**  def xyz(proj2):  if (proj2["label"] == 'Heart Rate'):  return proj2["vital\_reading"]  else :  return "0"  proj2['Heart\_Rate'] = proj2.apply(lambda proj2:xyz(proj2), axis=1)  **Temperature C:**  def xyz(proj2):  if (proj2["label"] == 'Temperature C'):  return proj2["vital\_reading"]  else :  return "0"  proj2['Temperature\_C'] = proj2.apply(lambda proj2:xyz(proj2), axis=1)  **Step 2:**  We have done some research on each Vital Reading and their Normal and Elevated values. According to the reference we have initialized the Criticality values for each of them using the below queries.  Find the Reference section for the documents which we took as reference.  **Systolic\_BP\_Range:**  def xyz(proj2\_rand\_samp) :    if (proj2\_rand\_samp["Systolic\_BP"] >= 120) & (proj2\_rand\_samp["Systolic\_BP"]<= 129):  return "1"  elif (proj2\_rand\_samp["Systolic\_BP"] >= 130) & (proj2\_rand\_samp["Systolic\_BP"]<= 139):  return "2"  elif (proj2\_rand\_samp["Systolic\_BP"] >= 140) & (proj2\_rand\_samp["Systolic\_BP"]<= 179):  return "3"  elif (proj2\_rand\_samp["Systolic\_BP"] >= 180):  return "4"  elif (proj2\_rand\_samp["Systolic\_BP"] < 105):  return "3"  else :  return "0"  proj2\_rand\_samp["Systolic\_BP\_range"] = proj2\_rand\_samp.apply(lambda proj2\_rand\_samp:xyz(proj2\_rand\_samp),  axis = 1)  **Diastolic\_BP\_range:**  def xyz(proj2\_rand\_samp) :    if (proj2\_rand\_samp["Diastolic\_BP"] >= 80) & (proj2\_rand\_samp["Diastolic\_BP"]<= 89):  return "1"  elif (proj2\_rand\_samp["Diastolic\_BP"] >= 90) & (proj2\_rand\_samp["Diastolic\_BP"]<= 120):  return "2"  elif (proj2\_rand\_samp["Diastolic\_BP"] >= 120):  return "3"  elif (proj2\_rand\_samp["Diastolic\_BP"] >= 73) & (proj2\_rand\_samp["Diastolic\_BP"]<= 80):  return "0"  elif (proj2\_rand\_samp["Diastolic\_BP"] < 73):  return "3"  else :  return "5"    proj2\_rand\_samp["Diastolic\_BP\_range"] = proj2\_rand\_samp.apply(lambda proj2\_rand\_samp:xyz(proj2\_rand\_samp),  axis = 1)  **Heart\_Rate\_Range:**  def xyz(proj2\_rand\_samp) :    if (proj2\_rand\_samp["age"] >= 20) & (proj2\_rand\_samp["age"]<= 29) & (proj2\_rand\_samp["Heart\_Rate"]>=100) & (proj2\_rand\_samp["Heart\_Rate"]<=170):  return "0"  elif (proj2\_rand\_samp["age"] >= 30) & (proj2\_rand\_samp["age"]<= 39) & (proj2\_rand\_samp["Heart\_Rate"]>=95) & (proj2\_rand\_samp["Heart\_Rate"]<=157):  return "0"  elif (proj2\_rand\_samp["age"] >= 40) & (proj2\_rand\_samp["age"]<= 49) & (proj2\_rand\_samp["Heart\_Rate"]>=90) & (proj2\_rand\_samp["Heart\_Rate"]<=149):  return "0"  elif (proj2\_rand\_samp["age"] >= 50) & (proj2\_rand\_samp["age"]<= 59) & (proj2\_rand\_samp["Heart\_Rate"]>=85) & (proj2\_rand\_samp["Heart\_Rate"]<=140):  return "0"  elif (proj2\_rand\_samp["age"] >= 60) & (proj2\_rand\_samp["age"]<= 70) & (proj2\_rand\_samp["Heart\_Rate"]>=80) & (proj2\_rand\_samp["Heart\_Rate"]<=128):  return "0"  else :  return "4"  proj2\_rand\_samp["Heart\_Rate\_range"] = proj2\_rand\_samp.apply(lambda proj2\_rand\_samp:xyz(proj2\_rand\_samp),  axis = 1)  **Respiratory Rate:**  def xyz(proj2\_rand\_samp) :    if (proj2\_rand\_samp["age"] >= 18) & (proj2\_rand\_samp["age"]<= 70) & (proj2\_rand\_samp["Respiratory\_Rate"]>=12) & (proj2\_rand\_samp["Respiratory\_Rate"]<=20):  return "0"  elif (proj2\_rand\_samp["age"] >= 70) & (proj2\_rand\_samp["Respiratory\_Rate"]>=15) & (proj2\_rand\_samp["Respiratory\_Rate"]<=20):  return "0"  else :  return "4"  proj2\_rand\_samp["Respiratory\_Rate\_range"] = proj2\_rand\_samp.apply(lambda proj2\_rand\_samp:xyz(proj2\_rand\_samp),  axis = 1)  **Oxygen\_Saturation:**  def xyz(proj2\_rand\_samp) :    if (proj2\_rand\_samp["Oxygen\_Saturation"] >= 95):  return "0"  elif (proj2\_rand\_samp["Oxygen\_Saturation"] >= 90) & (proj2\_rand\_samp["Oxygen\_Saturation"]<= 95):  return "2"  else :  return "4"    proj2\_rand\_samp["Oxygen\_Saturation\_range"] = proj2\_rand\_samp.apply(lambda proj2\_rand\_samp:xyz(proj2\_rand\_samp),  axis = 1)  **Temperature\_C:**  def xyz(proj2\_rand\_samp) :    if (proj2\_rand\_samp["Temperature\_C"] >= 35.1) & (proj2\_rand\_samp["Temperature\_C"]< 38):  return "0"  elif (proj2\_rand\_samp["Temperature\_C"] >= 38) & (proj2\_rand\_samp["Temperature\_C"]< 39.9):  return "2"  elif (proj2\_rand\_samp["Temperature\_C"] >= 39.9) & (proj2\_rand\_samp["Temperature\_C"]< 41):  return "3"  elif (proj2\_rand\_samp["Temperature\_C"] >= 41):  return "4"  elif (proj2\_rand\_samp["Temperature\_C"] < 35):  return "2"  else :  return "5"  proj2\_rand\_samp["Temperature\_C\_range"] = proj2\_rand\_samp.apply(lambda proj2\_rand\_samp:xyz(proj2\_rand\_samp),  axis = 1)  **Step 3:**  Next, we have created a column Criticality Range which is the sum of all the 6 vital readings which we created in Step 2.  proj2\_rand\_samp['Criticality\_Range'] = proj2\_rand\_samp.loc[:, 'Systolic\_BP\_range':'Temperature\_C\_range'].sum(1)  **Exploratory Data Analysis:**  **Average of Criticality Range for each Admission** Average of Criticality Range for each Admission Type.  Details are shown for Admission Type. **Details are shown for Admission Type.**  image  **AGE VS AVG of Criticality range** :    **Count of Age for each Label.  Size shows average of the Criticality Range.  The marks are labeled by average of Criticality Range.**  image  **Diagnosis, an average of Criticality Range and count of Hadm Id. Color shows details about the Admission Type. Size shows an average of Criticality Range. The marks are labeled by Diagnosis, an average of Criticality Range, and count of Hadm Id.d on Diagnosis and average of Criticality Range.**    **Admission Type, Diagnosis, average of Criticality Range and count of Criticality Range.  Color shows average of Criticality Range.  Size shows average of Criticality Range.**  image  From the EDA we found the below top 20 critical diagnosis types.  ['ALCOHOLIC HEPATITIS', 'SADDLE PE;TELEMETRY', '(AML) ACUTE MYELOGENOUS LEUKEMIA', 'ASCITIS',  'A,FLUTTER W,RVR,? NEW BRAIN MASS','CANCER','C3 FRACTURE',  'ATRIAL FIBRILLATION\PHARMACOLOGICAL / NON PHARM MGMT AFIB ABLATION \*\*REMOTE WEST\*\*',  'AORTIC STENOSIS\ AORTIC VALVE REPLACEMENT','ALCOHOL WITHDRAWAL','BLADDER RUPTURE;SEPSIS','ARTERIAL OCCLUSION',  'ASPIRATION PNEUMONIA','ABDOMINAL PERFORATION','ACUTE RESPIRATORY DISTRESS SYNDROME',  'ACUTE RENAL FAILURE;HYPOTENSION', 'ATRIAL TACHYCARDIA\EP STUDY; ATRIAL TACHYCARDIA ABLATION \*\*REMOTE WEST\*\*/SDA',  'ALTERED MENTAL STATUS;TELEMETRY', 'ASCENDING AORTIC ANEURYSM\BENTAL PROCEDURE REDO',  'ALTERED MENTAL STATUS;FEBRILE']  Accordingly, we have created a final criticality column based on the below key columns  **Populated Avg\_Criticality column based on the independent variables Admission\_Type, Diagnosis, Age, Criticality\_Range.**  def xyz(df) :    if (df["admission\_type"] == 'EMERGENCY') | (df["diagnosis"] == diagnosis) & (df["Criticality\_Range"] >3):  return "5"  elif (df["icustay\_id"] != 'NULL') & (df["Criticality\_Range"] >3):  return "4"  elif (df["admission\_type"] == 'EMERGENCY') & (df["Criticality\_Range"]<=3 ) & (df["Criticality\_Range"] > 1):  return "2"  elif (df["Criticality\_Range"] == 1):  return "1"  else:  return "0"    df["Avg\_Criticality"] = df.apply(lambda df:xyz(df),  axis = 1)  Azure Machine Learning Model:  For this project, we have used Azure Machine Learning which is a workspace where you create, build, and train machine learning models. It is a drag-and-drop tool (Azure Machine Learning Designer) where you can drag the data sets and further process the analysis of that data. It offers both no-code and low-code options for projects.  Building A Model:   * Create a blank project then drag the uploaded dataset into the experiment * Drag and drop the select columns in the dataset item after the data set * Next, we launched the column selector for the above item and select all the dependent and independent variables from our dataset * Now we add the split data item to the columns item and specify the train and test data split range * Then we connect the training data split from the above item to the train model item and add the machine learning model(Two-Class Boosted Decision Tree) to the train model * At this step, the model will be trained using the train data and specified machine learning model * Later we connect the output of the train model and the test data from the split data item to score model item to test how well our model is working on test data * Finally we add the evaluate model item and connect it to score model to view the accuracy and predictions   Graphical user interface  Description automatically generated  **Columns Selected**  Graphical user interface, application  Description automatically generated  **Score Model & Evaluate Results**  Table  Description automatically generated  Table  Description automatically generated  Results :  Discussions :  Conclusions :   * From our Model, we conclude that the model’s prediction accuracy is 84.5% * We would like to explore more to find underlying variables and useful insights to build the model to predict the Diagnosis/ ICD9 code used by doctors. * Also, we will need to build more different models to evaluate which model fits better.   Contributions :  Mounika Peddayearramreddygari:  discussed and explored several articles with reference to the project and composed the introduction, motivation, and conclusions of the project proposal. discussed and explored several data sets from the MIMIC-iii demo database using Big Query,From Admission Table - Filtered all the Diagnosis Types which Lead to Death. Found out which Diagnosis Types lead to how many Deaths by using Distinct Counts. We found there are 50 Diagnosis Types that led to death. Now did the same above but the diagnosis types did not lead to death and is not critical. But we saw that the same diagnosis types which had more deaths also lieat the top in fewer deaths. So, started exploring all other underlying factors that are involved in this by exploring more tables. Does this have anything to do with Admission Type? Either Emergency/Responsive. Checked and found out No. Collected the Subject\_ID/HADM\_id from Admission Table and linked what ICD Codes have been used to treat them from the diagnoses\_icd table. Each Subject had multiple ICD codes associated with in diagnoses\_icd table. And also checked for those ICD code descriptions in the d\_icd\_diagnoses table, EDA along with formulating complex SQL queries for data gathering and data cleaning using python. Worked on formulating complex SQL queries for data gathering and data cleaning using python.  Petchetti Naga Vijaya Lakshmi: discussed and explored several articles with reference to the project and composed the introduction, motivation, and conclusions of the project proposal. several data sets from the MIMIC-iii demo database using Big Query, worked on the ICU\_Stay table and its schema relationship tables to discover interesting patterns, EDA along with SQL queries for data gathering, SQL queries for data gathering and created a new column and identified the fields to find the criticality of the patient.  Bolisetti Sai Satwik: discussed and explored several articles with reference to the project and composed the introduction, motivation, and conclusions of the project proposal, several data sets from the MIMIC-iii demo database using Big Query, Set up the Google Big Query for the MIMIC-III database and started analyzing the tables by querying into the MIMIC\_clinical database and generating the relationship between tables like Admission Table, diagnoses\_icd table, d\_icd\_diagnoses, chartevents, d\_items, ICU\_Stay and working on data cleaning, worked on EDA along with formulating complex SQL queries for data gathering and data cleaning using python, formulating complex SQL queries for data gathering and data cleaning using python, standardizing the data and started researching on the different ML algorithms to create the ML model to identify the criticality of the patient.  Konatham Nikhil Reddy:  discussed and explored several articles with reference to the project and composed the introduction, motivation, and conclusions of the project proposal. discussed and explored several data sets from the MIMIC-iii demo database using Big Query, worked on data analysis and cleaning, and helped in documenting the relationship between the table for future reference. worked on EDA and data cleaning on python.  References :   * All the entity relationships and database details in this project are taken from the Mimic-iii schema and Entity Relationship on the below page. <https://mit-lcp.github.io/mimic-schema-spy/index.html> * The dataset can be found using the below U : <https://physionet.org/content/mimiciii/1.4/> * <https://www.health.state.mn.us/diseases/coronavirus/pulseoximeter.html> * <https://www.medicalnewstoday.com/articles/323819> * <https://www.researchgate.net/figure/The-Group-age-range-of-age-and-the-normal-Respiratory-rate-table-type-styles-4_tbl1_342236322> * <https://emoha.com/blogs/health/what-is-normal-bp-range-for-women-and-men>   https://emoha.com/blogs/health/what-is-normal-bp-range-for-women-and-men | | | | |  |