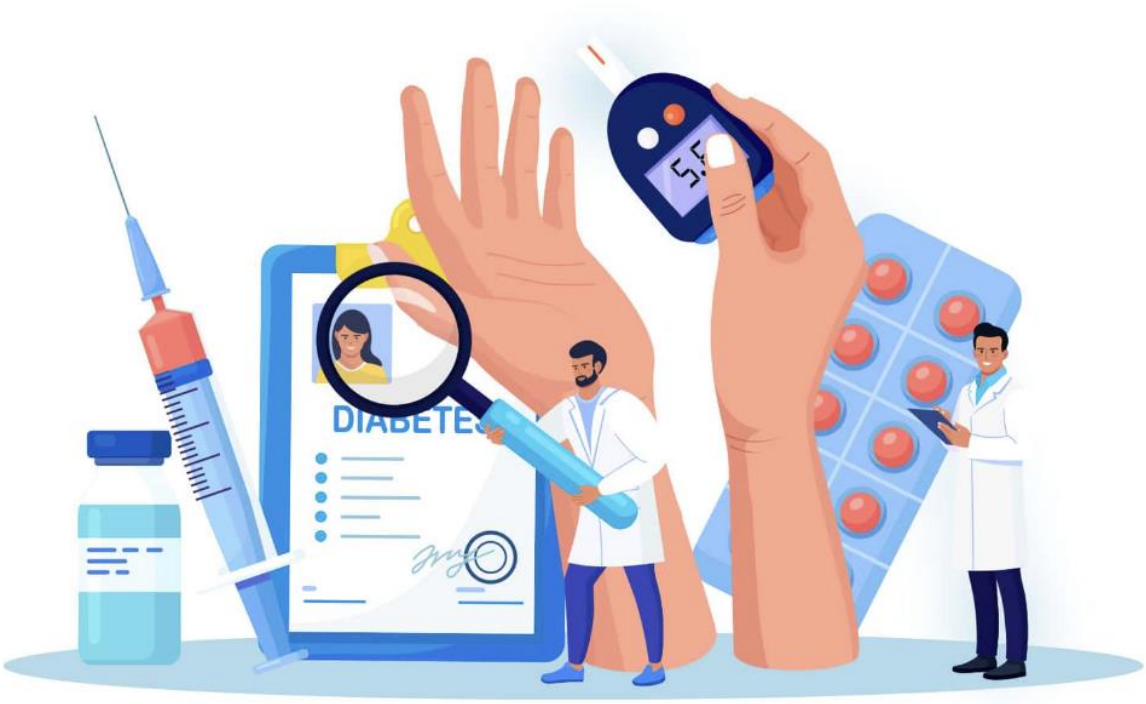
DTSC701/CSCI 636
INTRODUCTION TO BIG DATA
Group project: "Analyzing Diabetes Trends: A Comprehensive Study of Hospital Data (1999-2008) for Readmission Prediction".



Group members:

- **Oksana Dura 1316268**
- **Chinasa Okonkwo 1319774**
- **Akena Hutchinson 1269747**
- **Krishnarjuna Reddy Acholu 1319435**

Professor:

Liangwen Wu

Table of Contents

Abstract:

This group project focuses on the comprehensive analysis of a dataset spanning the years 1999 to 2008, encompassing information on diabetes across 130 hospitals. The dataset comprises 47 features, 101,766 instances, and presents challenges with missing values. The primary objective of our study is to uncover insightful patterns within the dataset. To achieve this, we engage in meticulous data preprocessing and subsequently employ a classification algorithm. Our specific aim is to predict the likelihood of patient readmission based on various data attributes, including but not limited to age, race, and medical history. Through this investigation, we seek to contribute valuable insights to the understanding of factors influencing diabetes-related hospital readmissions.

Snippet of the dataset:

There are 101766 rows and 50 columns.

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	payer_
50433	151518324	59097483	Caucasian	Male	[50-60)	?	1	7	7	4	
100197	422638592	42055272	Caucasian	Male	[80-90)	?	3	1	1	2	
59114	166708554	23195241	Caucasian	Male	[20-30)	?	1	1	7	1	
85904	272626374	68701014	AfricanAmerican	Male	[60-70)	?	1	1	7	6	
86387	274639074	54842598	Caucasian	Female	[30-40)	?	2	1	1	1	
87357	278896272	103224339	Caucasian	Female	[70-80)	?	2	6	7	6	
5851	29854332	84362391	Caucasian	Male	[70-80)	?	1	1	7	8	
53506	156927606	99106191	Caucasian	Female	[70-80)	?	1	6	7	5	
74539	221960250	41775885	Caucasian	Female	[80-90)	?	1	6	7	2	
46611	143700726	80966943	Caucasian	Male	[80-90)	?	1	6	7	4	

Original source of the data set https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008

Data Dictionary

- encounter_id: Unique identifier of an encounter.
- patient_nbr: Unique identifier of a patient.
- race: Race. Values are Caucasian, Asian, African American, Hispanic, and other.
- gender: Gender. Values: male, female, and unknown/invalid.
- age: Age Grouped in 10-year intervals: [0, 10), [10, 20),..., [90, 100).
- weight: Weight in pounds.
- admission_type_id: Integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, newborn, and not available.
- discharge_disposition_id: Integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available.

- admission_source_id: Integer identifier corresponding to 21 distinct values, for example, physician referral, emergency room, and transfer from a hospital.
- time_in_hospital: Integer number of days between admission and discharge.
- payer_code: Integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay.
- medical_specialty: Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family/general practice, and surgeon.
- num_lab_procedures: Number of lab tests performed during the encounter.
- num_procedures: Number of procedures (other than lab tests) performed during the encounter.
- num_medications: Number of distinct generic names administered during the encounter.
- number_outpatient: Number of outpatient visits of the patient in the year preceding the encounter.
- number_emergency: Number of emergency visits of the patient in the year preceding the encounter.
- number_inpatient: Number of inpatient visits of the patient in the year preceding the encounter.
- diag_1: The primary diagnosis (coded as first three digits of ICD9); 848 distinct values.
- diag_2: Secondary diagnosis (coded as first three digits of ICD9); 923 distinct values.
- diag_3: Additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct values.
- number_diagnoses: Number of diagnoses entered to the system.
- max_glu_serum. Indicates the range of the result or if the test was not taken. Values: >200, >300, normal, and none if not measured.
- A1Cresult: Indicates the range of the result or if the test was not taken. Values: >8 if the result was greater than 8%, >7 if the result was greater than 7% but less than 8%, normal if the result was less than 7%, and none if not measured.
- metformin: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- repaglinide: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- nateglinide: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- chlorpropamide: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter,

down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.

- glimepiride: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- acetohexamide. The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- glipizide: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- glyburide: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- tolbutamide: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- pioglitazone: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- rosiglitazone: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- acarbose: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- miglitol:The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- troglitazone: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- tolazamide: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter,

down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.

- examide: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- citoglipton: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- insulin: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- glyburide-metformin: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- glipizide-metformin: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- glimepiride-pioglitazone: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- metformin-rosiglitazone. The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- metformin-pioglitazone: The feature indicates whether the drug was prescribed or there was a change in the dosage. Values: up if the dosage was increased during the encounter, down if the dosage was decreased, steady if the dosage did not change, and no if the drug was not prescribed.
- change: Indicates if there was a change in diabetic medications (either dosage or generic name). Values: change and no change.
- diabetesMed: Indicates if there was any diabetic medication prescribed. Values: yes and no.
- readmitted: Days to inpatient readmission. Values: <30 if the patient was readmitted in less than 30 days, >30 if the patient was readmitted in more than 30 days, and No for no record of readmission

In this database, we have 3 different outputs:

- No readmission.
- A readmission in less than 30 days.
- A readmission in more than 30 days.

The purpose of this study is to find out indicators that a patient will be readmitted or not.

Development of the project (Setting up the environment):

Data Lake vs. Data warehouse

In our project focused on diabetes data analysis across 130 hospitals from 1999 to 2008, the choice of a data lake over a data warehouse is deliberate and strategic. Here's why:

- Data Diversity: Our dataset encompasses varied data types and formats. A data lake's schema flexibility allows us to store raw, unprocessed data without rigid structures.

- Scalability: Dealing with extensive datasets spanning multiple years and hospitals demands scalability. Data lakes, built on distributed storage, offer a cost-efficient solution for handling large volumes of data.

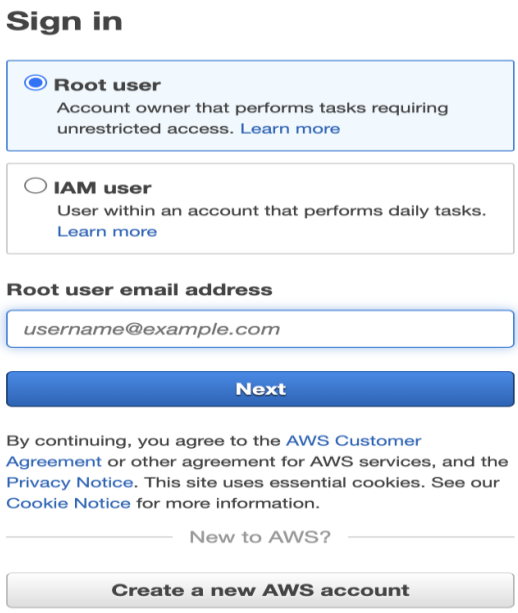
- Cost Efficiency: Storing raw data in its native format reduces preprocessing requirements, potentially lowering storage costs compared to traditional data warehouses.

- Advanced Analytics: Enabling direct access to raw data, data lakes facilitate advanced analytics and machine learning, crucial for our goal of uncovering patterns and predicting readmissions.

After choosing Data Lake the next steps require setting up EMR and S3. To successfully execute the program please follow the steps carefully:

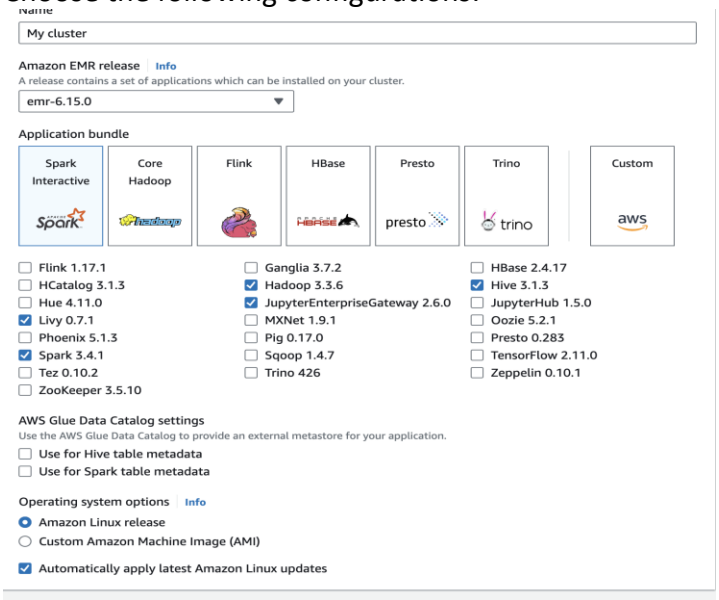
AWS Account:

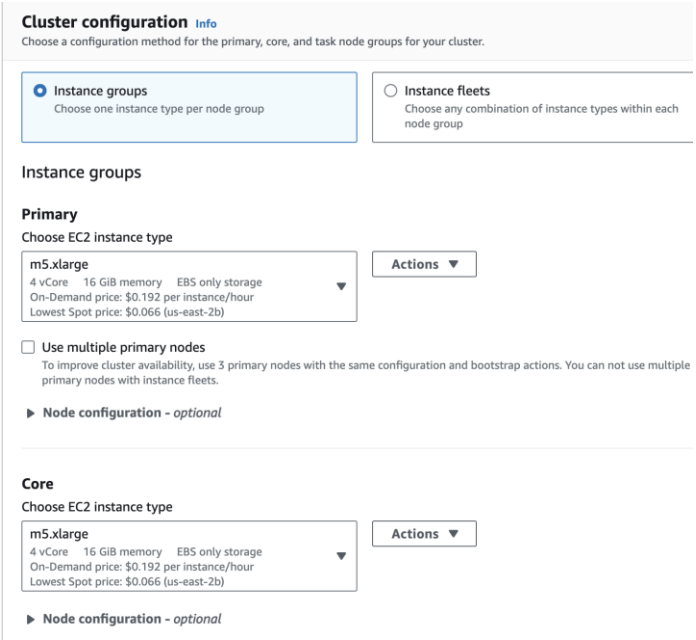
- Ensure you have an AWS account with the necessary permissions to create and manage EMR clusters.



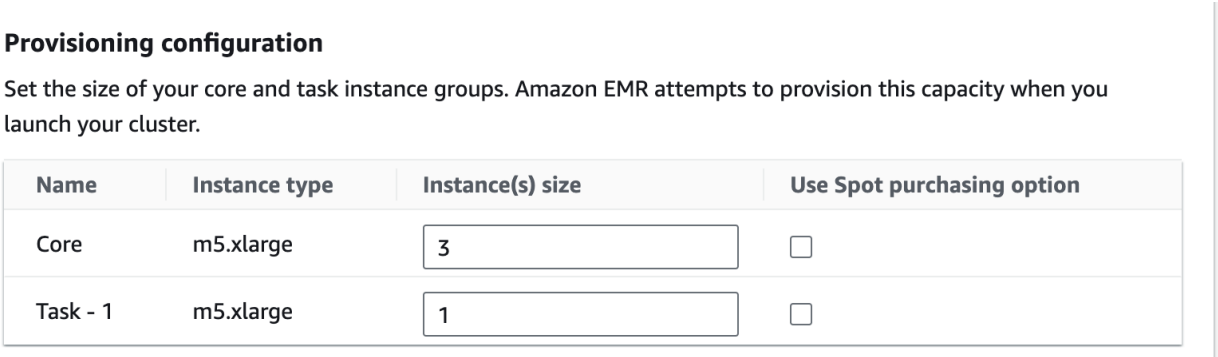
Create an EMR Cluster:

Choose the following configurations:

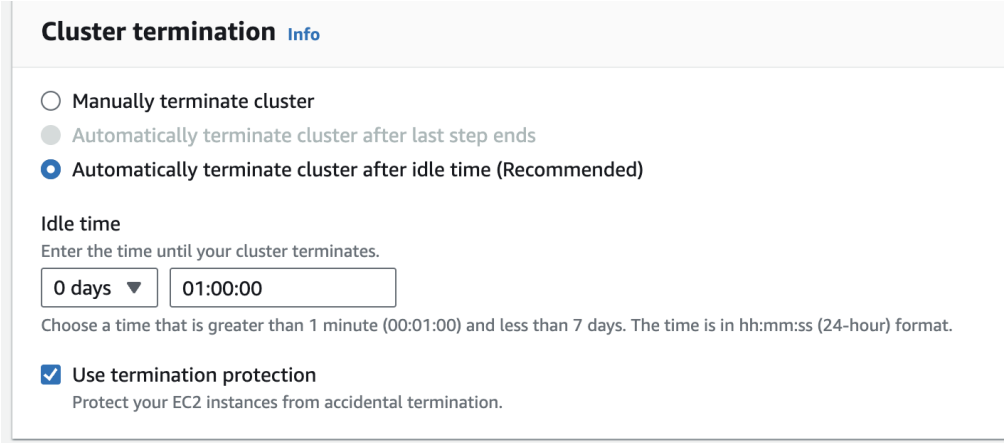




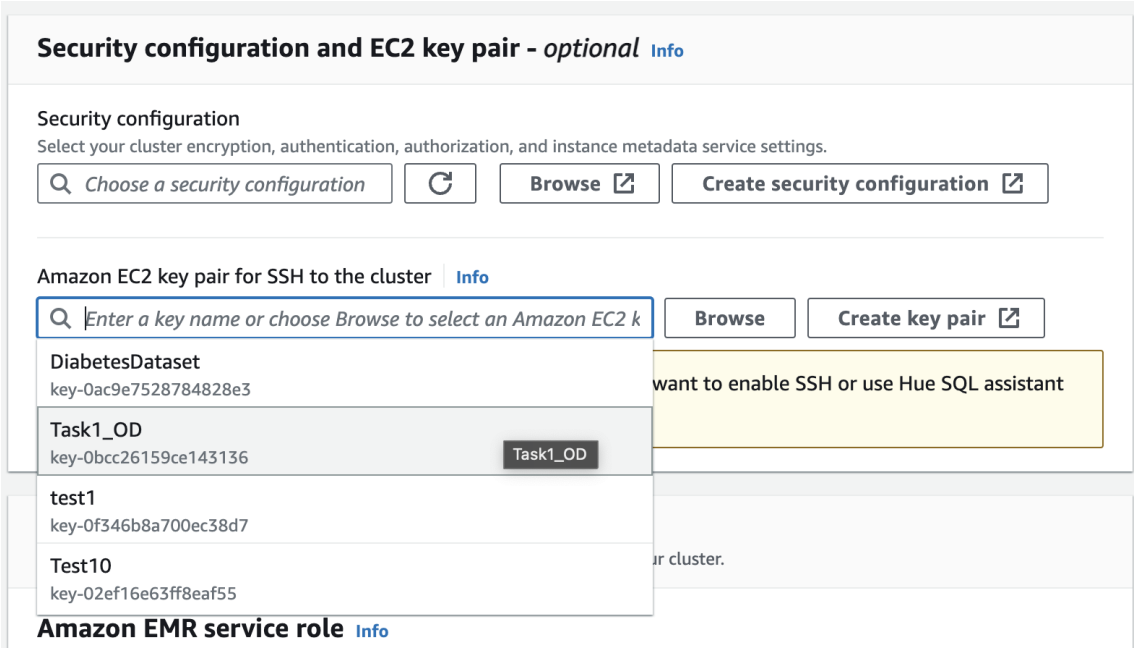
In Hardware Configuration for Instance type: m5.xlarge, Number of instance: 3



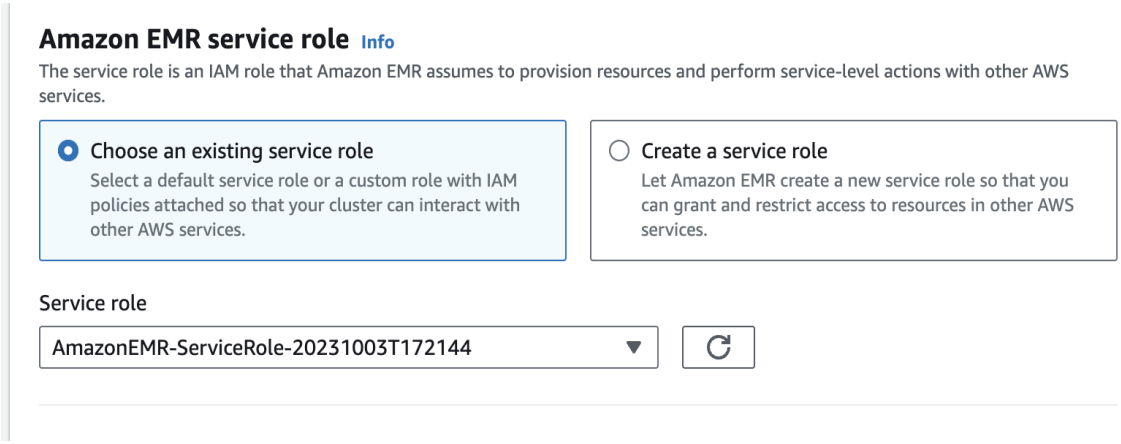
Decide how long your cluster with run for:



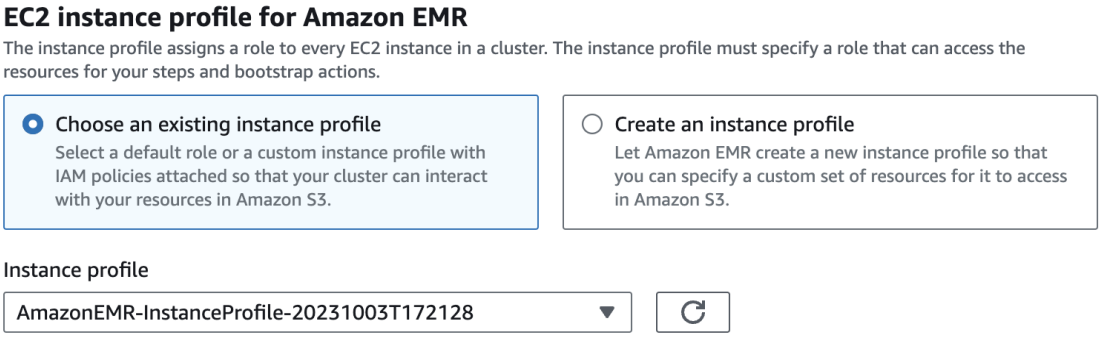
Create SSH key or use one of the options



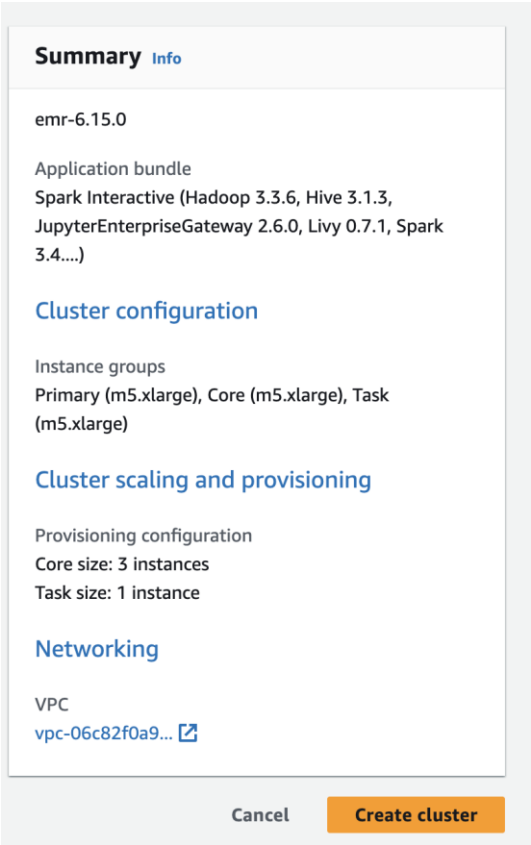
Choose the service role:



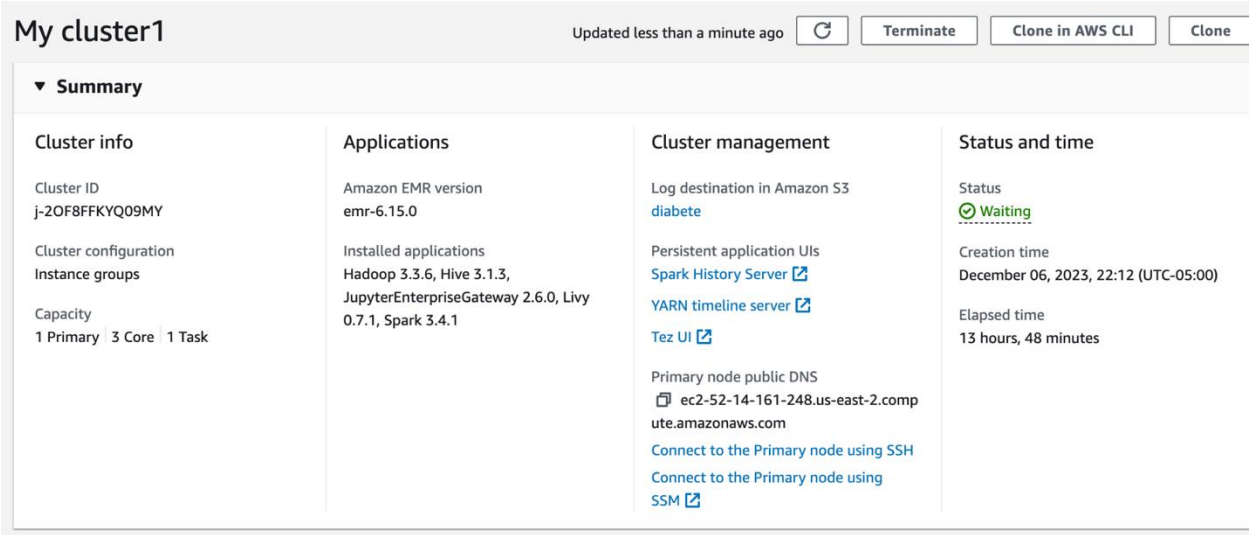
Choose the Instance Profile:



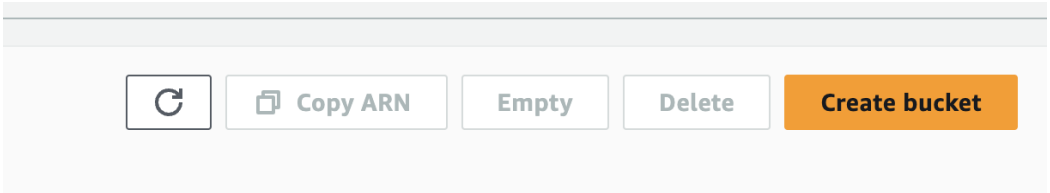
Create a cluster:



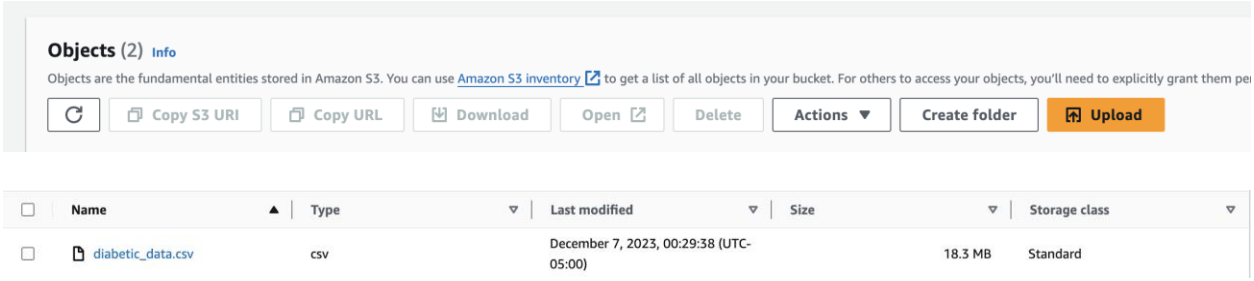
The cluster was created:



Go to the AWS S3 service. Create a bucket.



Upload the csv file:



Preprocessing the data with a spark application:

(The project was worked on a MAC terminal the commands might change for Windows or Linux operating system)

Open the terminal and connect to the cluster through the following commands:
Make sure the key has the right permissions. If key file has restrictive permissions, you might need to run chmod 400

```
Last login: Thu Dec  7 09:34:03 on ttys001
(base) oksanadura@Oksanas-MacBook-Pro ~ % chmod 400 /Users/oksanadura/Downloads/
Test10.pem
```

SSH into the Master Node:
After creating an EMR cluster, obtain the public DNS or IP address of the master node from the AWS EMR console.
Use SSH to connect to the master node

Then run the following command on the terminal:

vi preprocessing.py
Paste the preprocessing code here. First press I in the keyboard to be able to input data. Then to exit press esc and after :wq enter. Once you are out press the following command on your terminal:
spark-submit preprocessing.py --packages org.apache.hadoop:hadoop-aws:3.3.0

One the program is successful excecated the preprocessing file will be saved on S3 in the output folder.

☐	output/	Folder	-

Name	Type	Last modified	Size	Storage class
_SUCCESS	-	December 7, 2023, 01:46:43 (UTC-05:00)	0 B	Standard
part-00000-8e68a8c5-8e4d-4aad-a99b-fe7d9916a2f2-c000.csv	csv	December 7, 2023, 01:46:43 (UTC-05:00)	3.0 MB	Standard
part-00001-8e68a8c5-8e4d-4aad-a99b-fe7d9916a2f2-c000.csv	csv	December 7, 2023, 01:46:42 (UTC-05:00)	3.0 MB	Standard
part-00002-8e68a8c5-8e4d-4aad-a99b-fe7d9916a2f2-c000.csv	csv	December 7, 2023, 01:46:42 (UTC-05:00)	3.1 MB	Standard
part-00003-8e68a8c5-8e4d-4aad-a99b-fe7d9916a2f2-c000.csv	csv	December 7, 2023, 01:46:42 (UTC-05:00)	3.2 MB	Standard
part-00004-8e68a8c5-8e4d-4aad-a99b-fe7d9916a2f2-c000.csv	csv	December 7, 2023, 01:46:42 (UTC-05:00)	1.8 MB	Standard

High-level application history

Spark 3.4.1-amzn-2 **History Server**

Event log directory: hdfs:///var/log/spark/apps

Last updated: 2023-12-07 12:03:16

Client local time zone: America/New_York

Show 60 entries Search:

Version	App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.4.1-amzn-2	application_1701919193092_0027	DatasetPreprocessing	2023-12-07 01:45:25	2023-12-07 01:46:42	1.3 min	hadoop	2023-12-07 01:46:42	Download
3.4.1-amzn-2	application_1701919193092_0026	DatasetPreprocessing	2023-12-07 01:18:54	2023-12-07 01:19:24	29 s	hadoop	2023-12-07 01:19:24	Download
3.4.1-amzn-2	application_1701919193092_0025	DatasetPreprocessing	2023-12-07 01:16:31	2023-12-07 01:17:39	1.1 min	hadoop	2023-12-07 01:17:40	Download
3.4.1-amzn-2	application_1701919193092_0024	DatasetPreprocessing	2023-12-07 01:12:22	2023-12-07 01:13:31	1.2 min	hadoop	2023-12-07 01:13:31	Download
3.4.1-amzn-2	application_1701919193092_0023	DatasetPreprocessing	2023-12-07 01:05:35	2023-12-07 01:06:12	37 s	hadoop	2023-12-07 01:06:12	Download
3.4.1-amzn-2	application_1701919193092_0022	DatasetPreprocessing	2023-12-07 01:00:44	2023-12-07 01:01:19	35 s	hadoop	2023-12-07 01:01:19	Download
3.4.1-amzn-2	application_1701919193092_0021	DatasetPreprocessing	2023-12-07 00:57:09	2023-12-07 00:57:39	30 s	hadoop	2023-12-07 00:57:39	Download
3.4.1-amzn-2	application_1701919193092_0020	DatasetPreprocessing	2023-12-07 00:50:38	2023-12-07 00:51:10	31 s	hadoop	2023-12-07 00:51:10	Download
3.4.1-amzn-2	application_1701919193092_0019	DatasetPreprocessing	2023-12-07 00:44:03	2023-12-07 00:44:30	27 s	hadoop	2023-12-07 00:44:30	Download
3.4.1-amzn-2	application_1701919193092_0018	DatasetPreprocessing	2023-12-07 00:39:35	2023-12-07 00:40:07	31 s	hadoop	2023-12-07 00:40:07	Download
3.4.1-amzn-2	application_1701919193092_0017	DatasetPreprocessing	2023-12-07 00:34:59	2023-12-07 00:35:57	58 s	hadoop	2023-12-07 00:35:57	Download
3.4.1-amzn-2	application_1701919193092_0016	DatasetPreprocessing	2023-12-07 00:30:42	2023-12-07 00:31:38	56 s	hadoop	2023-12-07 00:31:38	Download
3.4.1-amzn-2	application_1701919193092_0015	DatasetPreprocessing	2023-12-07 00:26:26	2023-12-07 00:26:55	30 s	hadoop	2023-12-07 00:26:55	Download
3.4.1-amzn-2	application_1701919193092_0014	DatasetPreprocessing	2023-12-07 00:23:31	2023-12-07 00:24:00	29 s	hadoop	2023-12-07 00:24:00	Download

Preprocessing Spark Application code explained.

This Python script utilizes PySpark, a Python library for Apache Spark, to preprocess and save a dataset related to diabetes.

- **Importing Libraries:**
  The script begins by importing necessary PySpark and machine learning libraries.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.sql import functions as F
from pyspark.sql.types import DoubleType
from pyspark.sql.functions import when
from pyspark.ml.feature import StringIndexer, OneHotEncoder
from pyspark.ml import Pipeline
# Assemble the features into a vector
from pyspark.ml.feature import VectorAssembler
```

- **Spark Session Setup:**
  It creates a Spark session named "DatasetPreprocessing" to interact with the Spark cluster.

- **Loading the Dataset:**

```
# Load the dataset from S3
df = spark.read.csv(input_path, header=True, inferSchema=True)
```

- **Data Cleaning:**

```
# Columns to drop
columns_to_drop = ['discharge_disposition_id','admission_source_id', 'number_inpatient','encounter_id','number_emergency',
                   'patient_nbr', 'weight', 'payer_code', 'medical_specialty']
df = df.drop(*columns_to_drop)


# Define the value you want to replace
value_to_replace = '?'
# Loop through all columns in the DataFrame
for column in df.columns:
# Replace the specified value with None
    df = df.withColumn(column, when(df[column] == value_to_replace, None).otherwise(df[column]))


#Filling missing values in the 'race' column with the most frequent value
most_frequent_race = df.groupby("race").count().orderBy(col("count").desc()).select("race").first()[0]
df = df.withColumn("race", F.when(col("race").isNull(), most_frequent_race).otherwise(col("race")))

# Dropping rows with any remaining missing values
df = df.dropna()
```

- **Diagnosis Columns Preprocessing:**

```python
#diag_1, diag_2, and diag_3 columns to preprocess
columns_to_preprocess = ['diag_1', 'diag_2', 'diag_3']

# Step 1: Replace non-numeric values with NaN
for col_name in columns_to_preprocess:
    non_numeric_condition = ~(col(col_name).cast("double").isNotNull())
    df = df.withColumn(col_name, F.when(non_numeric_condition, None).otherwise(col(col_name)))

# Step 2: Convert to Numeric
for col_name in columns_to_preprocess:
    df = df.withColumn(col_name, col(col_name).cast(DoubleType()))

# Drop rows with NaN (adjust as needed)
df = df.dropna(subset=columns_to_preprocess)
```

- **Categorical Column Encoding**

```python
categorical_columns = [ 'max_glu_serum', 'A1Cresult', 'metformin', 'repaglinide',
                        'nateglinide', 'chlorpropamide', 'glimepiride', 'acetohexamide', 'glipizide',
                        'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol',
                        'troglitazone', 'tolazamide', 'examide', 'citoglipton', 'insulin',
                        'glyburide-metformin', 'glipizide-metformin', 'glimepiride-pioglitazone',
                        'metformin-rosiglitazone', 'metformin-pioglitazone', 'change', 'diabetesMed']

# Use StringIndexer to convert categorical values to numerical indices
indexers = [StringIndexer(inputCol=col, outputCol=col+"_index", handleInvalid="keep") for col in categorical_columns]

# Use OneHotEncoder to one-hot encode the categorical indices
encoders = [OneHotEncoder(inputCol=col+"_index", outputCol=col+"_encoded") for col in categorical_columns]

# Map the target variable 'readmitted' to numerical values
label_indexer = StringIndexer(inputCol="readmitted", outputCol="label")

# Define the feature columns
feature_columns = [col+"_encoded" for col in categorical_columns]
```

Pipeline Creation / Execution

```python
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")

# Define the stages of the pipeline
stages = indexers + encoders + [label_indexer, assembler]

# Create a pipeline
pipeline = Pipeline(stages=stages)

# Fit the pipeline to transform the data
transformed_data = pipeline.fit(df).transform(df)



# Save the preprocessed dataset back to S3
transformed_data.write.format("csv").mode("overwrite").option("header", "true").save(output_path)
```

- **Error Handling:**

```python
def preprocess_and_save_data(input_path, output_path):
    try: ...

    except Exception as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    # Define input and output paths
    input_path = "s3://diabete/diabetic_data.csv"
    output_path = "s3://diabete/output/"

    # Call the function
    preprocess_and_save_data(input_path, output_path)
```

Decision tree model (The model was executed on Google Collab)

Preparing the data for testing and training:

```python
from sklearn.model_selection import train_test_split

# Assuming 'data_encoded' is your DataFrame with the encoded features and target variable
X = data_encoded.drop('readmitted', axis=1)  # Features
y = data_encoded['readmitted']  # Target variable

# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Display the shapes of the resulting sets
print("Training set - X:", X_train.shape, "y:", y_train.shape)
print("Testing set - X:", X_test.shape, "y:", y_test.shape)
```

Define the hyperparameters to search through for Decision Tree

```python
criterion_DT = ["gini", "entropy"]
splitter_DT = ["best", "random"]
max_features_DT = ["sqrt", "log2", None]
max_depth_DT = [2,4,6,8,10,12]
```

Initialize variables to store the best parameters and accuracy

```
highest_accuracy_DT = 0
f = ""
g = ""
h = ""
y = ""
```

Loop through hyperparameters and train Decision Tree models

```
for y in max_depth_DT:
 for x in max_features_DT:
   for n in criterion_DT:
     for z in splitter_DT:
       clf = DecisionTreeClassifier(criterion = n, max_features = x, splitter= z, max_depth = y,random_state = 101)
       clf.fit(X_train,y_train)
       spam_test_target_predict=clf.predict(X_test)
       print("For criterion_DT = ",n,", and max_features_DT = ", x , "and splitter_DT = ", z, "max_depth_DT = ", y, " the accuracy score is: ", accuracy_score(y_

       if accuracy_score(y_test,spam_test_target_predict) > highest_accuracy_DT:
         highest_accuracy_DT = accuracy_score(y_test,spam_test_target_predict)
         f = n
         g = x
         h = z
         w = y
```

Train the best Decision Tree model

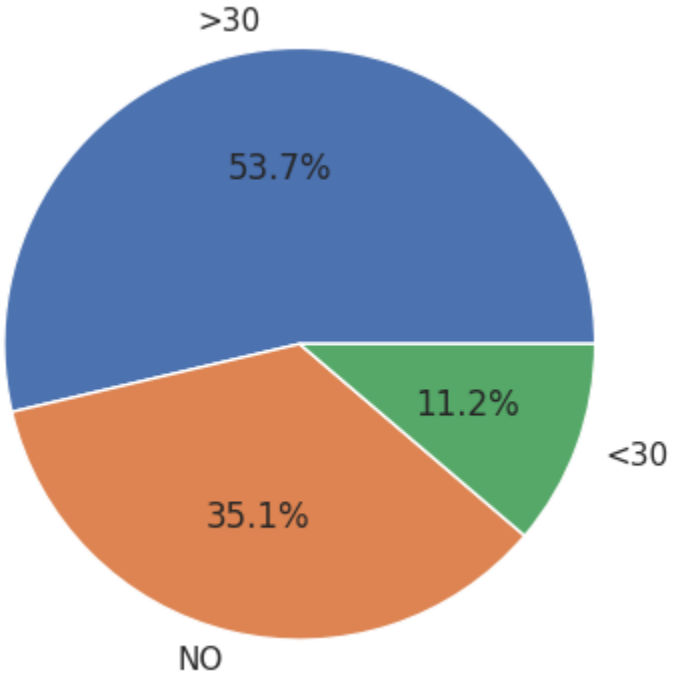
```
# Train the best Decision Tree model
clf = DecisionTreeClassifier(criterion = f, max_features = g, splitter = h, random_state = 101,max_depth = y )
clf.fit(X_train,y_train)

# Make predictions on the test set using the best model
spam_test_target_predict=clf.predict(X_test)

# Evaluate the model using confusion matrix, classification report, and accuracy score
c_m = confusion_matrix(y_test,spam_test_target_predict)
c_r = classification_report(y_test,spam_test_target_predict)
a_s = accuracy_score(y_test,spam_test_target_predict)
```

Output:

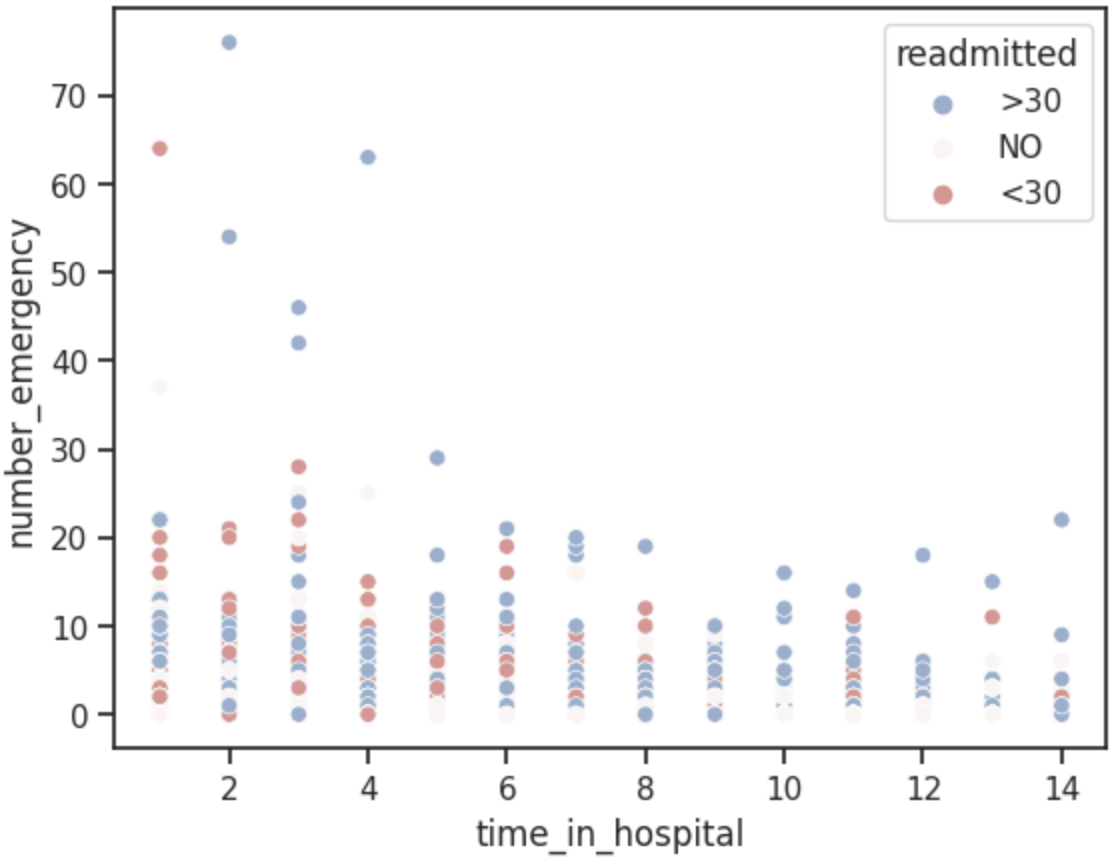
```
For criterion_DT =  gini , and max_features_DT =  None and splitter_DT =  random max_depth_DT =  12  the accuracy score is:  0.6207574012600773
For criterion_DT =  entropy , and max_features_DT =  None and splitter_DT =  best max_depth_DT =  12  the accuracy score is:  0.6136440620554163
For criterion_DT =  entropy , and max_features_DT =  None and splitter_DT =  random max_depth_DT =  12  the accuracy score is:  0.6206219090847503
The decision tree with the highest accuracy  0.6347130953187453 has the following parameters:
criterion_DT =  gini max_depth_DT =  12  max_features_DT =  None splitter_DT =  best
Prediction for 20 observation:     [0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 0 0 0 1 0]
Actual values for 20 observation:  [1 0 1 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 1 1]
[[5560 2578]
 [3143 3480]]
              precision    recall  f1-score   support

           0       0.64      0.68      0.66      8138
           1       0.57      0.53      0.55      6623

    accuracy                           0.61     14761
   macro avg       0.61      0.60      0.60     14761
weighted avg       0.61      0.61      0.61     14761

0.6124246324774745
```

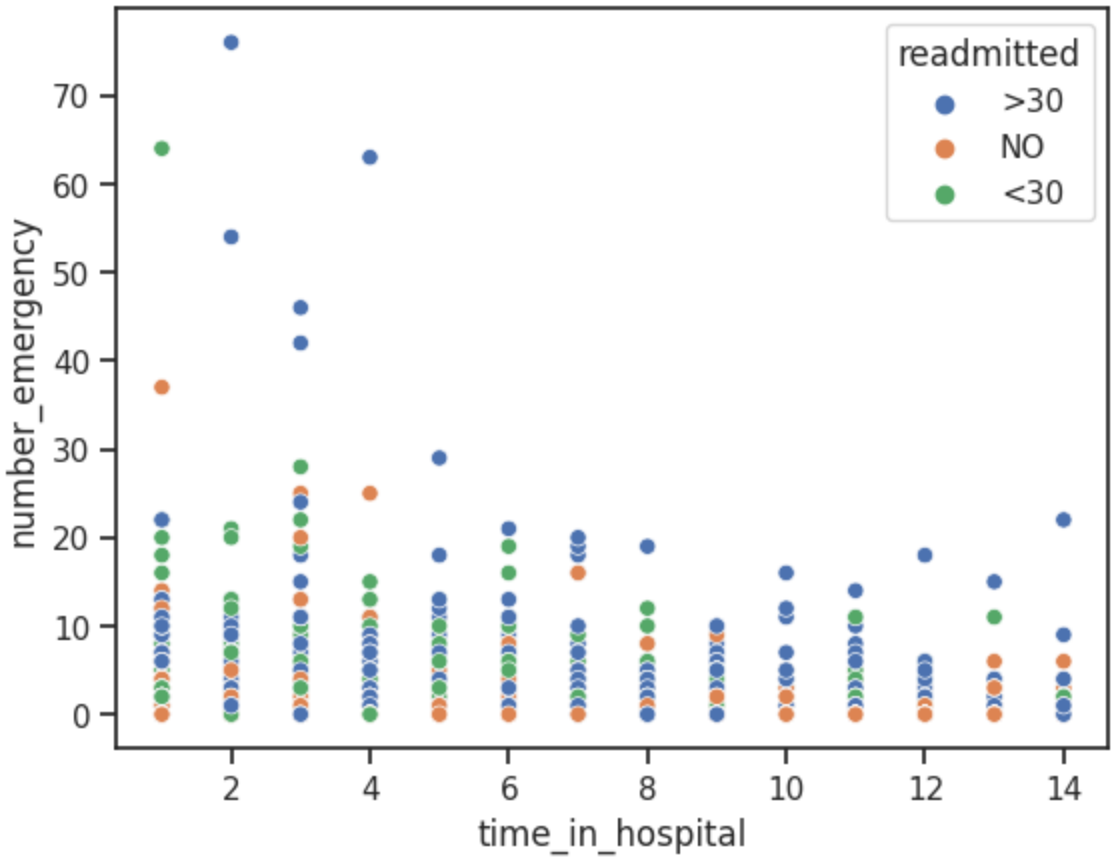
Graphical Analysis:

Percentage of patients who have been readmitted within 30 days or more

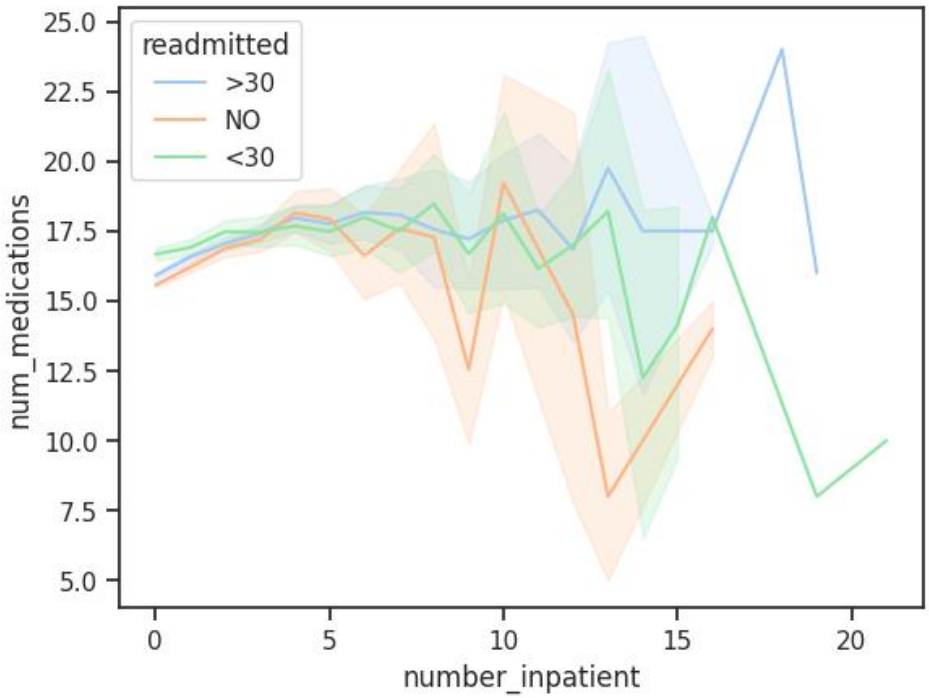


Time spent in the hospital by the number of emergency incidents. Each item is further separated by if or when the patient was readmitted to the hospital.

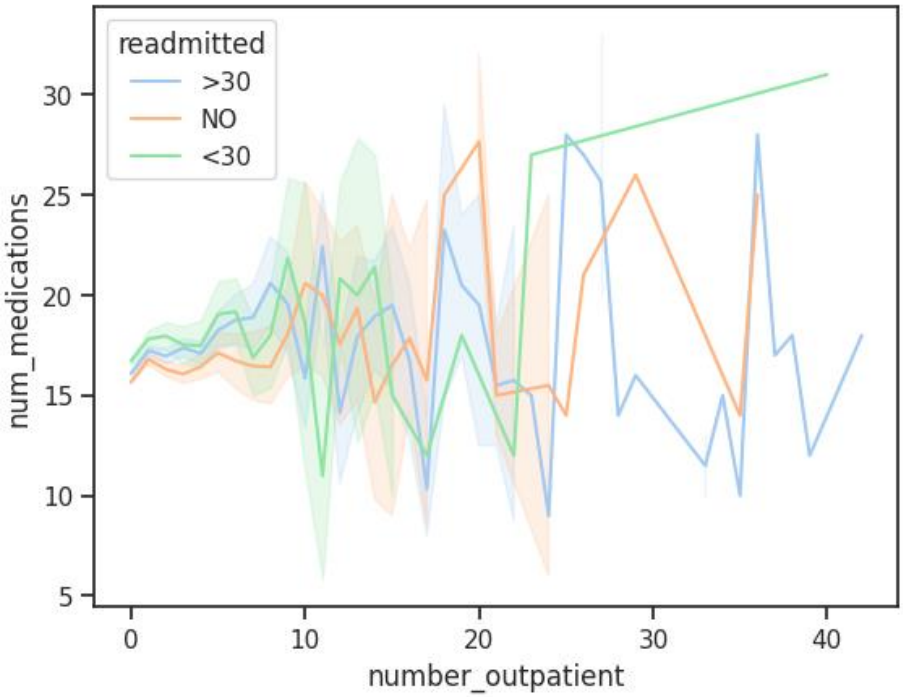

readmitted
>30
NO
<30
number_emergency
0
10
20
30
40
50
60
70
time_in_hospital
2
4
6
8
10
12
14



Number of medications given to inpatients with a separation on when or if they have been readmitted to the hospital



Number of medications given to outpatients with a separation on when or if they have been readmitted to the hospital



Conclusion:

Confusion Matrix:		Actual Values	
		Positive	Negative
Predicted Values	Positive	5560	2578
	Negative	3143	3480

Accuracy: $\frac{\#correctly\ predicted\ values}{\#total\ amount\ of\ prediction} = \frac{9040}{14761}$ = 0.6124 = 61%

Error Rate: $\frac{\#incorrectly\ predicted\ values}{\#total\ amount\ of\ prediction} = \frac{5721}{14761}$ = 0.388 = 39%

Sensitivity: $\frac{\#correctly\ predicted\ as\ true}{\#actually\ label\ as\ true} = \frac{5560}{8703}$ = 0.638 = 64%

Specificity: $\frac{\#correctly\ predicted\ as\ false}{\#actually\ label\ as\ false} = \frac{3480}{6058}$ = 0.574 = 57%

Our prediction accuracy is about 61%. With a sensitivity rate of 64% and a specificity rate of 57% we can conclude that the current treatment plan may not be the most effective treatment to be given to the patients. More often than not patients are being readmitted into the hospital. Doctors may either have to consider outside factors such as diet and exercise into treatment as well as medication.