Udacity Nanodegree

Machine Learning Engineer

Capstone Project Report

“Predicting mortality in the Emergency Department utilizing Deep Learning Methods”

Student: Porfirio Basaldúa González

Project definition

Domain background

The time is ripe for a technological revolution in healthcare. According to Mckinsey, over the last decade pharmaceutical companies have been compiling research and development data into datasets, while “the US federal government and other public stakeholders have been opening their vast stores of health-care knowledge, including data from clinical trials and information on patients covered under public insurance programs” (Mckinsey & Company, 2013)
All this new influx of information has already been applied by the different stakeholders to create savings and improve processes, for example “Kaiser Permanente has fully implemented a new computer system [...] and achieved an estimated $1 billion in savings from reduced office visits and lab tests” (Mckinsey & Company, 2013). The same stakeholders acknowledge that there are still many opportunities for innovation and improvement within the healthcare domain, Mckinsey & Company estimate that “since 2010, more than 200 new businesses have developed innovative health-care applications. About 40 percent of these were aimed at direct health interventions or predictive capabilities.”

Of these possible innovations, predictive capabilities are of special interest to health care professionals, since it will allow them to “interrupt the patient’s [projected] trajectory and set him on the proper course” (Adamson, D., 2015, p. 30) this capability is known as Prescriptive Analytics, and is considered a “Holy Grail” (Adamson, D., 2015, p. 30) within the healthcare domain.

This project will aim to create a tool that can assist the physicians in obtaining such a predictive capability of the patient’s outcome, utilizing a supervised Machine Learning model to derive predictions from a relational database of different medical measurements from patients visits to the Emergency Department.

Dataset

This project will use the Medical Information Mart for Intensive Care (MIMIC) dataset. According to Johnson, A. et al., this dataset “is a large, freely-available database comprising de-identified health-related data from patients who were admitted to the critical care units of the Beth Israel Deaconess Medical Center.” It contains data from 2008 to 2019 collected from Metavision bedside monitors and is arranged in different relational tables.

The authors recognize the sensibility of the information contained in the dataset and therefore request that access to the dataset be conditioned to the following requisites:
1. Completing a training course in human subjects research.
2. Signing the data use agreement. Adherence to the terms of this agreement is paramount.
3. Accessing the data directly in the cloud.

Because of these considerations, the original dataset will not be provided as part of this project, interested readers can access the complete dataset by following the instructions described in Alistair, J., 2014.

It is important to note that the dataset consists of five modules (hosp, icu, ed, cxr, note) of which only the “ed” module will be used in this project. This module contains a relational database detailing medical measurements, administrative records, personal patient information (de-identified) and other details concerning individual admissions to the Emergency Department.
The database will be accessed through Google Cloud Platform, transformed into a denormalized tabular format and stored in AWS, where the implementation of the Machine Learning algorithm will be made.

Problem statement

Can patient healthcare records predict the likelihood of mortality in an emergency room visit?

Solution statement

Machine Learning models combined with large amounts of data are well suited for prediction problems, and cloud infrastructure will ensure the system has enough resources to handle big-data concerns effectively. With this information, the proposed solution is as follows:

Using cloud infrastructure, the researcher will create a data pipeline that will expose an API through the internet which will allow the healthcare professionals to access real time predictions from a machine learning model. This predictions will indicate if the patient is likely to die while in emergency care, giving health care professionals a tool for decision making, shortening the time needed for the patient to receive adequate care.

Metrics

Li C. et al. (2021) have done a similar research, comparing the performance of various models to assess the probability of death of a patient, using the same dataset as proposed in this project. Their investigation concludes that a LightGBM model is the best performer of the models they tested, with an accuracy of 93.6% for early mortality predictions in emergency departments, this will be used as a benchmark.

This project will implement the same model, allowing the researcher to directly compare the result of this project with that of the cited paper, which will act as a benchmark. Conversely, the metric used in this project will be the accuracy of the model.

Project Analysis

Data exploration

The image below shows the overall characteristics of the data used in this project, some important considerations are:

The extracted data consists of a table of 205503 rows, each representing one patient visit to the Emergency Department. The table is stored in a pandas dataframe, built from the CSV file that was constructed in GCP and stored in S3 before beginning work on these steps.
There are 15 columns, most of them containing numerical information. The columns with a Dtype of object contain text-based information.
Columns number 7 to 14 contain null values, this missing information will have to be managed in further steps.
Column 0, labeled as 'deceased', contains the target that the machine learning model will predict. It is a binary column that is filled with 1 if the patient died, and 0 if the patient did not die.

```
RangeIndex: 205504 entries, 0 to 205503
Data columns (total 15 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   deceased               205504 non-null  int64
 1   subject_id             205504 non-null  int64
 2   gender                 205504 non-null  int64
 3   race                   205504 non-null  object
 4   arrival_transport      205504 non-null  object
 5   existing_doses         205504 non-null  int64
 6   medicine_dispensations 205504 non-null  int64
 7   temperature            191025 non-null  float64
 8   heartrate              194174 non-null  float64
 9   resprate               192573 non-null  float64
 10  o2sat                  192534 non-null  float64
 11  sbp                    193632 non-null  float64
 12  dbp                    193241 non-null  float64
 13  pain                   197462 non-null  object
 14  acuity                 200428 non-null  float64
```

Of the columns that are missing values, the 'temperature' column is the one with most missing data, with 7.05% percentage of blank values. As there are less than 10% missing inputs, the rows without data in all the columns will be discarded.

Inspecting other numerical columns, it is apparent that there are some erroneous values in the dataframe. For example, the maximum values for the vital sign measurements (temperature, heartrate, resprate, o2sat, sbp, dbp) are orders of magnitude above the limit pointed by Sapra et al., 2022, these outliers will be dropped as they can skew the results of the ML model. The 'medicine_dispensations' and 'existing_doses' have outliers, but is less clear if these are results of human error, because there is no objective reference to compare

them to. These values will be kept and they will be scaled so that the algorithm can work with them. The previous analysis is derived from the following table:

	existing_doses	medicine_dispensations	temperature	heartrate	resprate	o2sat	sbp	dbp	acuity
count	205504.000000	205504.000000	191025.000000	194174.000000	192573.000000	192534.000000	193632.000000	193241.000000	200428.000000
mean	14.536661	7.717869	98.026788	84.680572	17.414518	98.535019	135.276324	79.803498	2.627038
std	50.213646	19.452057	4.295442	17.956342	2.461268	9.550241	347.338749	154.066896	0.726921
min	0.000000	0.000000	0.300000	1.000000	0.000000	0.000000	1.000000	0.000000	1.000000
25%	0.000000	1.000000	97.600000	72.000000	16.000000	98.000000	120.000000	69.000000	2.000000
50%	3.000000	3.000000	98.000000	83.000000	18.000000	99.000000	132.000000	78.000000	3.000000
75%	12.000000	8.000000	98.600000	95.000000	18.000000	100.000000	147.000000	87.000000	3.000000
max	3958.000000	1437.000000	986.000000	1109.000000	209.000000	2100.000000	151103.000000	51989.000000	5.000000

The numerical columns can be input directly into the model, but further processing will be needed for the text-based columns to be understood by the algorithm.

The encoding techniques used to process the text-based columns into numbers were one-hot encoding and binary encoding.

In one-hot encoded columns, a separate column is added for every possible value in the original column and the rows of this new column are filled with 1 if the original value corresponds with the title of the new column, and 0 otherwise. This is the encoding schema used for the arrival_transport column, which gathers a few methods of transportation that the patients used to enter the hospital.

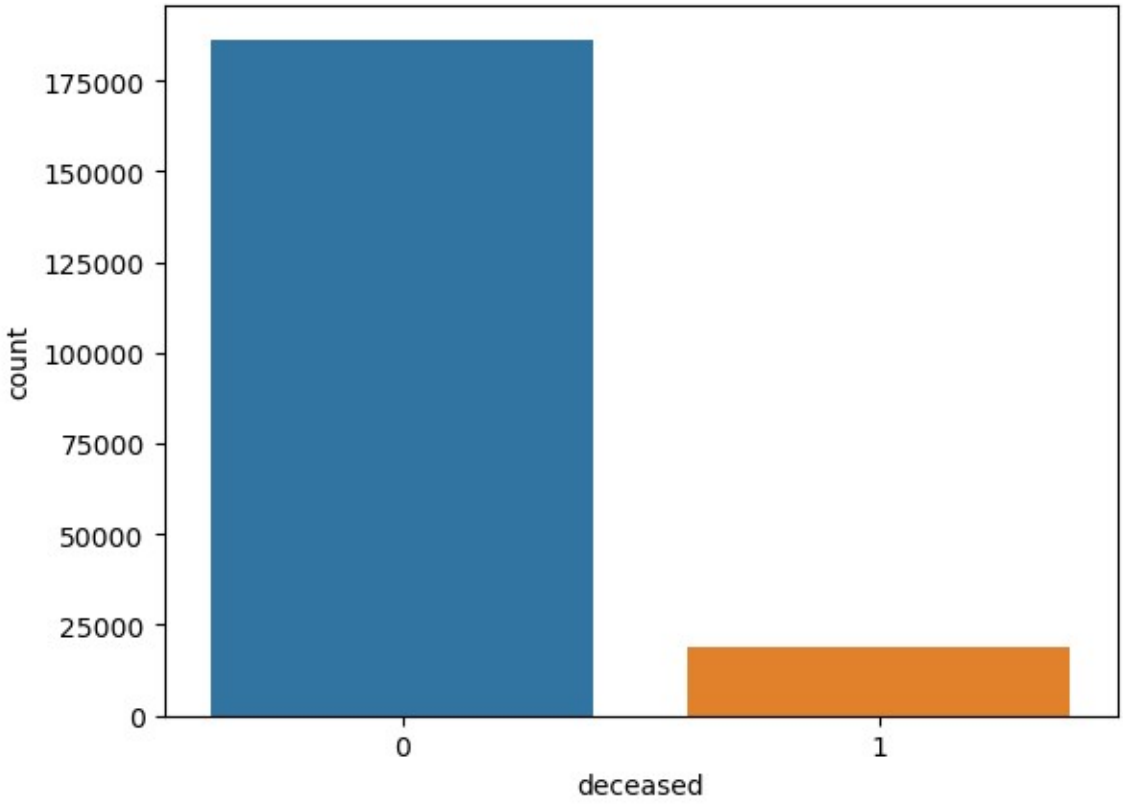
One hot encoders can create large sparse matrices if applied to columns with a lot of possible values. An alternative is to use binary encoding, where all the possible values of a column are assigned a binary-coded number, starting with 1. Next, every digit of that coded number is added as a separate column and appended to the dataframe. This is the encoding used for the ‘race’ column, transforming the categorical data into numbers.

The ‘pain’ column is described as a numerical column containing numbers from 0 to 10 that state the self-reported level of pain felt by the patients when entering the Emergency Department. Upon inspection, the column contains a wide range of numerical and text-based values, some of which cannot be readily interpreted as a pain scale, therefore this column is dropped from the table.

The subject_id column in the table identifies a row of the table as a single patient, and is used in the original MIMIC-IV dataset to connect the ED database with tables in other modules. It is not useful for the current analysis, and will be dropped from the dataframe.

Exploratory visualization

A defining characteristic of this data is that the target column is heavily skewed toward the class 0, which represents no mortality for that patient. When splitting the data, it is important that the random splits have samples of the class 1, so that the model is trained to recognize the characteristics of this class.
The following plot is a visualization of the total counts for each class in the target column.



Algorithms and techniques

The LightGBM algorithm is an implementation of a tree classification algorithm called Gradient Boost Decision Tree, this algorithm is renowed for its performance in ranking and multi-class classification problem uses an ensemble of trees that scan all the values of each feature to find the best split places, and then build a tree that can be used to predict new information. LightGBM is an implementation that seeks to reduce the number of scans that the algorithm has to conduct while building the tree, leading to faster model training times and less use of memory. As all GBDT implementations, LightGBM trees grow with each column (feature) and their values, therefore the model is not well suited to work with high-dimensionality data (many columns) or a high volume of values (a lot of rows)

Given the characteristics of the algorithm mentioned above, it is a good fit for the problem statement given, as well as the input data. This is a binary classification task that aims to predict if a patient will die (class 1) or not (class 0), furthermore, the input data has a low

number of features (columns) and few values for each feature, even being able to be processed in a personal computer.

As a widely recognized algorithm, LightGBM is supported in AWS Sagemaker as a built-in algorithm, having dedicated Docker images for training and deployment of models. As is the case in this project, AWS Sagemaker also supports custom implementation of this model, allowing the researcher to fine tune hyperparameters of the model as well as adding support tools tools such as distributed training of the model, and specific Docker images to train and deploy estimators with this algorithm (Mishra, A., 2019).

Once the model was selected and trained, the researcher can optimize the model and improve the model performance. In this project, the optimization technique that was used is hyperparameter tuning, which consists of many continuous iterations of training and evaluating the performance of the model, changing the hyperparameters of the model in each iteration and at the end selecting the combination of hyperparameters that yielded the best model performance. The Sagemaker platform supports many strategies for automatic selection of hyperparameters, the strategy used in this project is bayesian optimization, where the platform treats hyperparameter tuning like a Machine Learning regression problem, looking to optimize the performance of the built model with a set of hyperparameters provided by the user. Using tools such as specialized (Docker) images for training as well as distributed parallel jobs, the platform selects and reports the best-performing combination of hyperparameters for the proposed model. (Misra, A., 2018)

Benchmark

Accuracy refers to the closeness of a given set of predictions to the set of true values. The function used in this project calculates the multilabel classification accuracy, with the following definition (Cournapeau, D. et al., 2010):

If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the fraction of correct predictions over n_{samples} is defined as

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

where $1(x)$ is the indicator function.

Project Methodology

Data preprocessing

As stated in the analysis section, several actions have to be taken to clean the data and convert the raw values into numbers that the LightGBM algorithm can ingest to predict mortality for new patients. The activities are divided into the following:

1. Drop unused columns. The column ‘subject_id’ has no information that relates to the target prediction, therefore it is eliminated. The column ‘pain’ is not constructed as the authors described, having too many erroneous values it is hard to interpret, therefore it is unusable and will be eliminated from the final dataframe.

2. Drop null values. As stated in the analysis section, there are few values missing (less than 10% of the total values), therefore is possible to eliminate these incorrect inputs without losing a large volume of information.

3. Encode categorical variables. One-hot encoding is a common technique for handling the conversion of categories into numerical columns, it is a simple conversion done to the ‘arrival_transport’ column and it doesn’t create too many additional columns, as the number of choices is only 5. In contrast, there are more than 30 possible choices for the ‘race’ column, if these were encoded using one-hot encoders, the number of columns would double. Since the algorithm works better with fewer columns, a binary encoder was used to transform the ‘race’ column into 6 additional columns, that map a choice of race into a binary-encoded number.

4. Feature scaling. Because of the wide range of values in the columns ‘existing_doses’ and ‘medical_dispensations’, the algorithm can be skewed towards higher values, therefore these columns are scaled so that the values are between 1 and 0.

5. Drop erroneous readings. Some measurements are outside the scale of normal human vital sign readings, therefore these values are eliminated from the dataset.

Implementation

Once the dataframe is ready to be used as input for the Machine Learning model, the steps followed to implement the LightGBM model are as follows:

1. Train a LightGBM model in script mode, this gives freedom to chose the hyperparameters that will be modified, it also permits the user to input the custom dataframe created in the preprocessing step, this introduces more complexity for the developer, given the fact that a separate Python script is built to guide the Sagemaker platform in the correct training and further deployment of the model.

2. Create an API endpoint that will serve predictions from the previously trained model, it is an standard URL that can be accessed from the internet, and is backed by other AWS resources

that allow it to scale. Having an endpoint leaves the door open for future extensions and integrations with future projects.

While implementing these steps, the only complication was that the dedicated image for LightGBM training and deployment didn't work with the script mode and custom implementation of the LightGBM model, thus there was a need to substitute this image for a PyTorch estimator, adding the libraries needed for LightGBM. This is not ideal, because additional space is dedicated to PyTorch and related libraries, which can increase the size of the solution and introduce additional complications if future updates of the image lead to incompatibilities with LightGBM.

Furthermore, this step was coded after the process described in the refinement part, because in Sagemaker it is possible to conduct hyperparameter tuning and directly train a predictive model with the best combination of hyperparameter, thus having the best performing model for inference faster. Completing the refinement first has the added advantage of reducing resources allocated for training. In this project, since the dataframe is small, the training was made in a single machine, without the need for distributed training to look for the best model configuration.

Refinement

This steps were developed before the training and deployment of the model, for the reasons described above. The purpose of a hyperparameter search is to optimize the performance of the model and the Sagemaker platform automates the bulk of the process, improving the productivity of the developer.

The first step to conduct an automated search is to be familiar with the hyperparameters of the chosen model and how they affect performance, next is to define the hyperparameters that will be modified and the search space that the machine will explore, this is done with a HyperparameterRange dictionary.
Then, a distributed tuning job was launched, where the dataframe was replicated in 2 machines, each one given a subset of the parameters to test.
The tuning job automatically tests combinations of the hyperparameters, looping through a train then test cycle and tracking the best performance of the model. Finally, the platform reports the set of hyperparamenters that lead to the best performance, which is defined as the optimization of the metric that was defined in this project.

When the tuning is complete, an optimal version of the model can be trained and then used to predict new data, as described in the previous step.

Project Results

Model evaluation and validation

The accuracy was 91.4%, being lower than the benchmark metric of 93.6%. These numbers can be compared directly, since the authors of the research paper used the same model and dataset as this project.

The performance of the model developed here was inferior to the benchmark, but the difference is 2% of the benchmark value, which points to a satisfactory performance of the model developed in this project.
Perhaps the difference is explained by the feature engineering that was done in this project, and the limitation of the data that was considered for this project.

Future improvement

There are other databases in the MIMIC-IV dataset that contain information about other aspects of the patients medical procedures, these datasets can be included in future extensions of the project to extract other possible features, such as transfers to other units within the hospital, they can also be used to engineer new features, such as combining the complete history of visits for each patient, in addition of the first visit used in this project.

Further information includes periodic vital sign measurements, which can be used to conduct a time-series analysis and attempt predictions in real time, while the patient is undergoing medical treatment in the ED. This can be done as another project

Another improvement to be made is to automate the ETL pipeline described in this model, using big query and multicloud integration between GCP and AWS for a production level infrastructure deployment

Future projects can include the creation of an UI that will enable health care professionals to use the predictions in real situations.

References

Adamson, D. (2015, August 13). Big Data in Healthcare Made Simple: Where It Stands Today and Where It's Going. Retrieved March 12, 2023, from https://www.slideshare.net/healthcatalyst1/big-data-in-healthcare-made-simple-where-it-stands-today-and-where-its-going

Cournapeau, D., Brucher, M., Pedregosa, F., Varoquaux, G., Gramfort, A., & Michael, V. (2010). *3.3. metrics and scoring: Quantifying the quality of predictions*. scikit-learn. Retrieved April 3, 2023, from https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score

Huang, X. (2022, June 21). Amazon_Tabular_Classification_LightGBM_CatBoost.ipynb. Python source file. Dallas, Texas; Github. From https://github.com/aws/amazon-sagemaker-examples/blob/main/introduction_to_amazon_algorithms/lightgbm_catboost_tabular/Amazon_Tabular_Classification_LightGBM_CatBoost.ipynb

Johnson, A., Bulgarelli, L., Pollard, T., Horng, S., Celi, L. A., & Mark, R. (2021). MIMIC-IV (version 1.0). PhysioNet. https://doi.org/10.13026/s6n6-xd98.

Johnson, A. et al. (2014, August 25). *Getting started*. MIMIC. Retrieved April 3, 2023, from https://mimic.mit.edu/docs/gettingstarted/

Kayyali, B., Knott, D., & Kuiken, S. V. (2013, April 1). *The Big-Data Revolution in US health care: Accelerating value and innovation*. McKinsey & Company. Retrieved March 12, 2023, from https://www.mckinsey.com/industries/healthcare/our-insights/the-big-data-revolution-in-us-health-care#/

Mishra, A. (2018). *Bayesian Optimization*. Amazon. Retrieved April 3, 2023, from https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-how-it-works.html#automatic-tuning-bayesian-optimization.title

Mishra, A. (2019). *Machine learning in the AWS cloud: Add intellegence to applications with Amazon Sagemaker and Amazon Rekognition*. Amazon. Retrieved April 3, 2023, from https://docs.aws.amazon.com/sagemaker/latest/dg/lightgbm-HowItWorks.html

Ke, G. (2016). *Python-package introduction*. Python-package Introduction - LightGBM 3.3.5.99 documentation. Retrieved April 3, 2023, from https://lightgbm.readthedocs.io/en/latest/Python-Intro.html

Li, C., Zhang, Z., Ren, Y., Nie, H., Lei, Y., Qiu, H., Xu, Z., & Pu, X. (2021). Machine learning based early mortality prediction in the emergency department. *International Journal of Medical Informatics*, *155*, 104570. https://doi.org/10.1016/j.ijmedinf.2021.104570

Sapra, A., Malik, A., & Bhandari, P. (2022, May 8). *Vital Sign Assessment*. National Library of Medicine. Retrieved April 3, 2023, from https://www.ncbi.nlm.nih.gov/books/NBK553213/