MLDL homework

By Kar Ng

Literature Review and References

- Most of the time and effort in this homework was spent on research for relevant and interesting literature, Github projects, and datasets
- Python coding is modest since I run most of the analysis on the low-code platform PyCaret.
- I select a low-code platform for this homework since I believe that Python expertise is not the main object of this homework.

Dataset

- https://github.com/DanielSola/mimic-iii-project

- This is a MIMIC III dataset that contains a lot more ICU patient information than the MIMIC tables at MIT's QueryBuilder, especially physiological data (vital signs) and lab results.
- I use PyCaret to perform Regression, Classification, and Clustering analyses.
- The Jupyter notebooks are largely based on tutorials provided by PyCaret. I modified these tutorials for my selected dataset.

Features for Analysis

- Literature review shows that Lengths of Stay and Admission are two very popular measures for medical data analysis. But we are not permitted to analyze these for this homework.
- Hence, I look for datasets that provide information on other interesting measures like mortality, severity, and disease diagnosis.
- Moreover, I need datasets that also provide rich demographic, vital signs, and laboratory test data.
- The dataset at https://github.com/DanielSola/mimic-iii-project met most of these criteria.

Selection of Analysis Approaches

- My literature survey indicates that classification and regression are significantly more commonly used for healthcare statistical analysis.
- Typically, regression is used to target numeric measures like severity scores and mortality
- Classification is used to predict categorical variables.
- Clustering analysis seems less common, but it is still useful to identify potential healthcare issues (like epidemic) that occur in some demographic groups and/ or geographic locations.

Contrastive Learning using MedClip

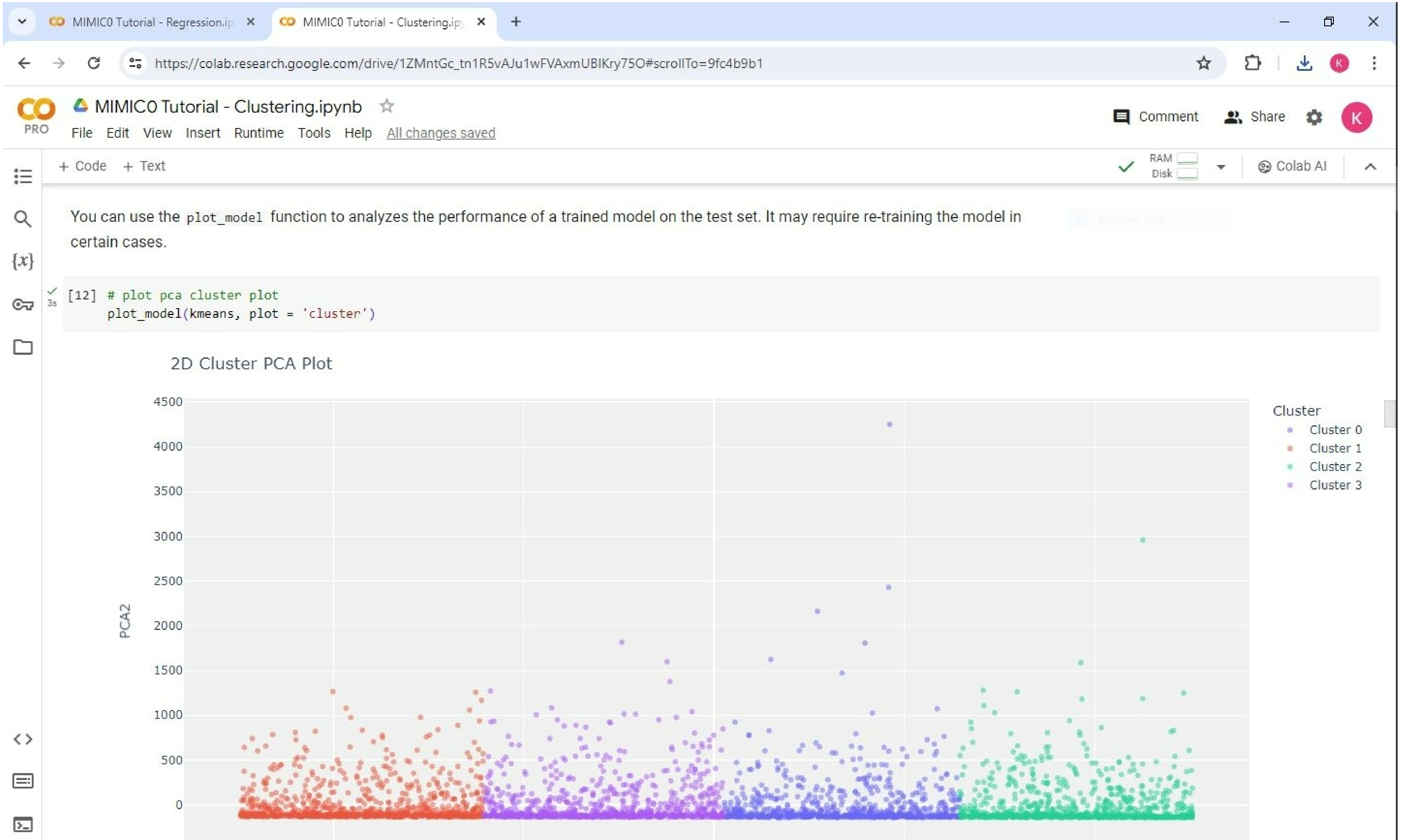
- For the bonus point, I discuss the use of MedClip for contrastive learning of diagnostic images.

Clustering

- I begin with a discussion of the Clustering analysis, since it is the shortest
- See the source code and results in the Jupyter notebook at
- https://colab.research.google.com/drive/1ZMntGc_tn1R5vAJu1wFVAxmUBIKry75O?usp=sharing
-   The K-Means methods is used. It identifies four clusters but the lack of a sharp elbow indicates that the algorithm used cannot clearly separate the four clusters.

PCA plot

- A principal component analysis plot shows similarities between groups of samples in a data set. Each point represents a correlation between an initial variable and the first and second principal components. https://builtin.com/data-science/step-step-explanation-principal-component-analysis



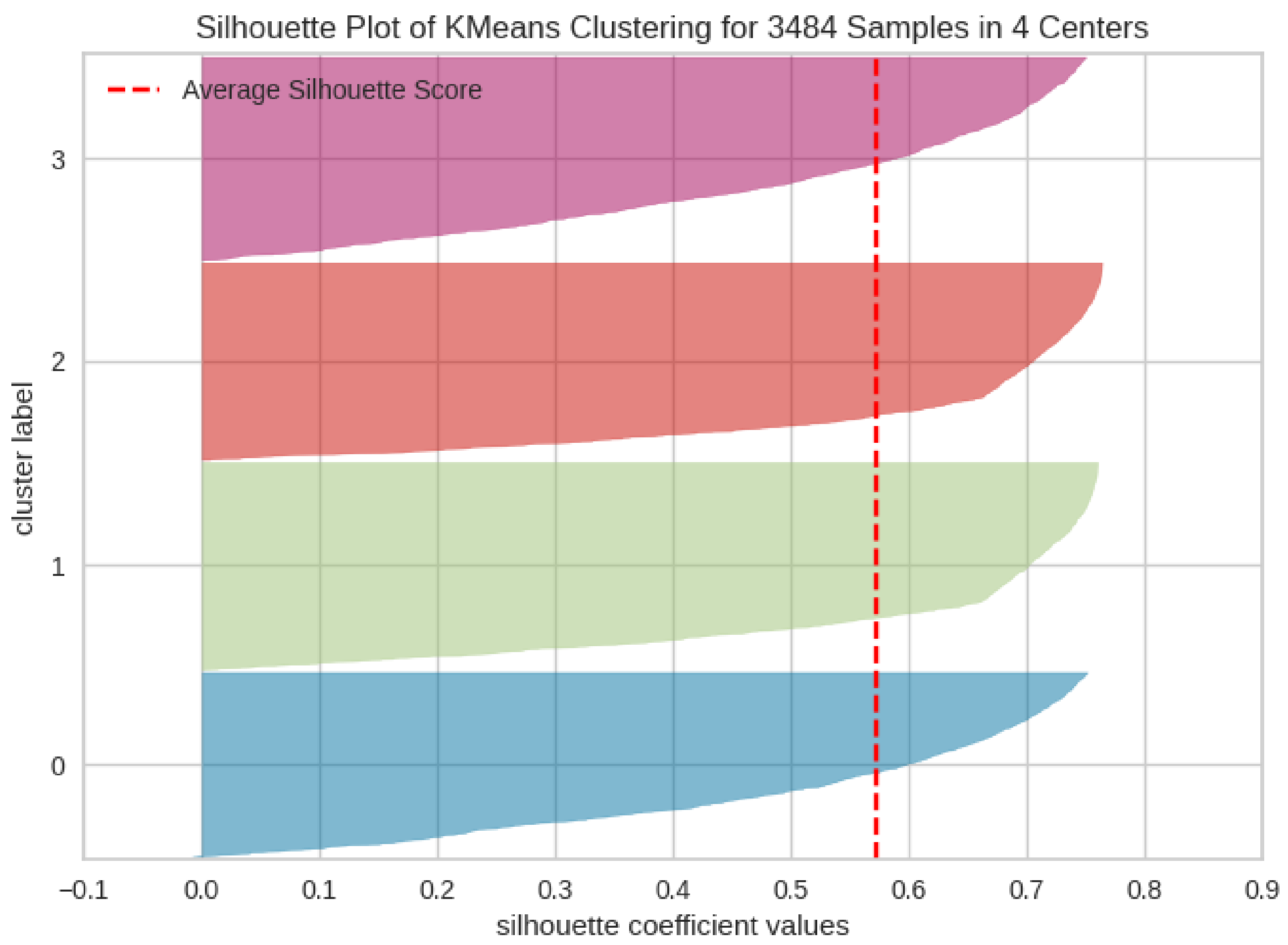
MIMIC0 Tutorial - Clustering.ipynb

You can use the `plot_model` function to analyzes the performance of a trained model on the test set. It may require re-training the model in certain cases.

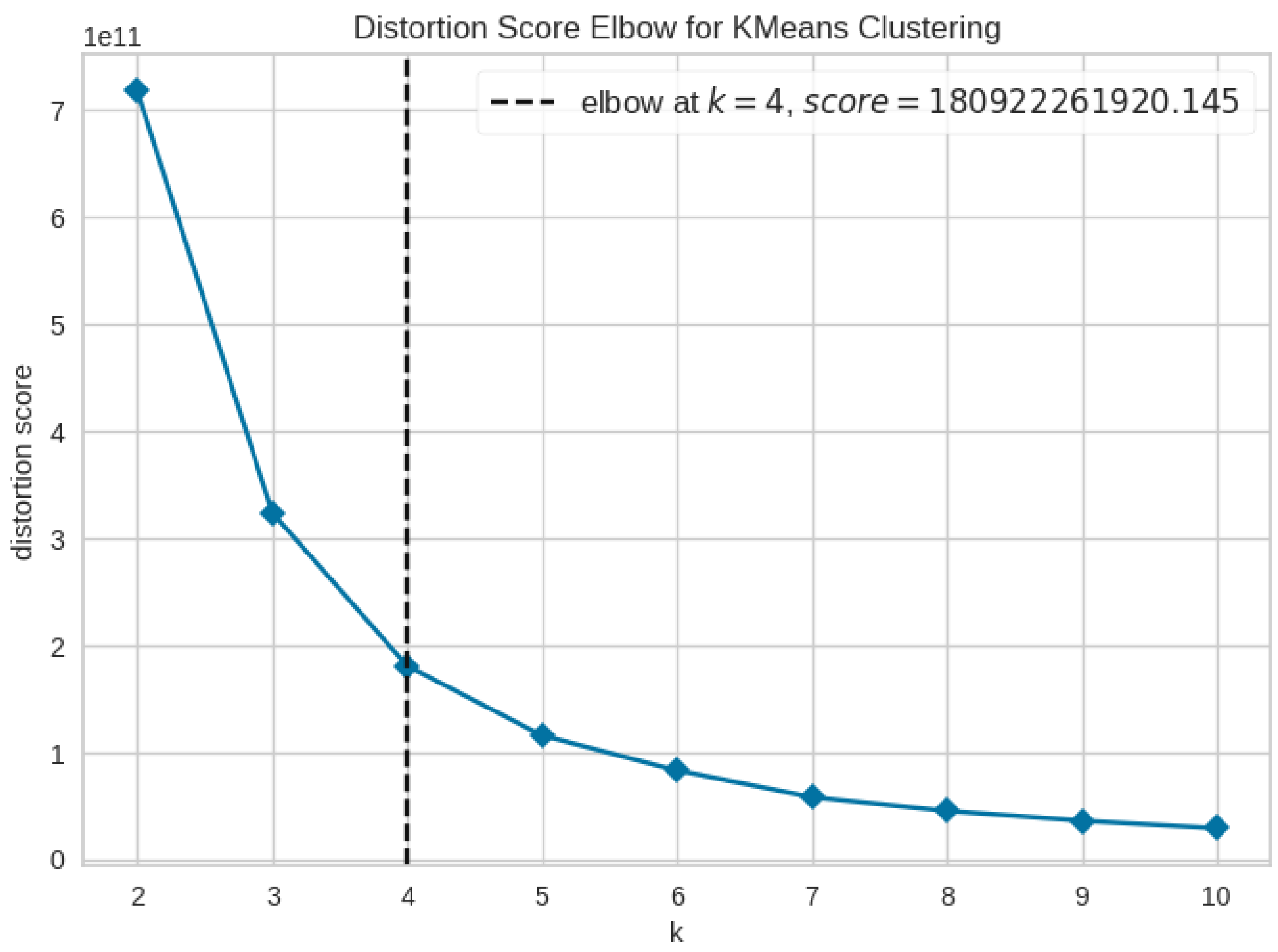
```
[12] # plot pca cluster plot
     plot_model(kmeans, plot = 'cluster')
```

2D Cluster PCA Plot

The silhouette coefficient or silhouette score (0 to 1 which is the best) is a **measure of how similar a data point is within-cluster (cohesion) compared to other clusters**



distortion score is the **average of the squared distances from the cluster centers of the** clusters.



Regression Analysis

- https://colab.research.google.com/drive/17xUqHJamUyoLkZxgTpuzo9SNfZw-dW6V?usp=sharing contains the Python code and results of my Regression analysis.

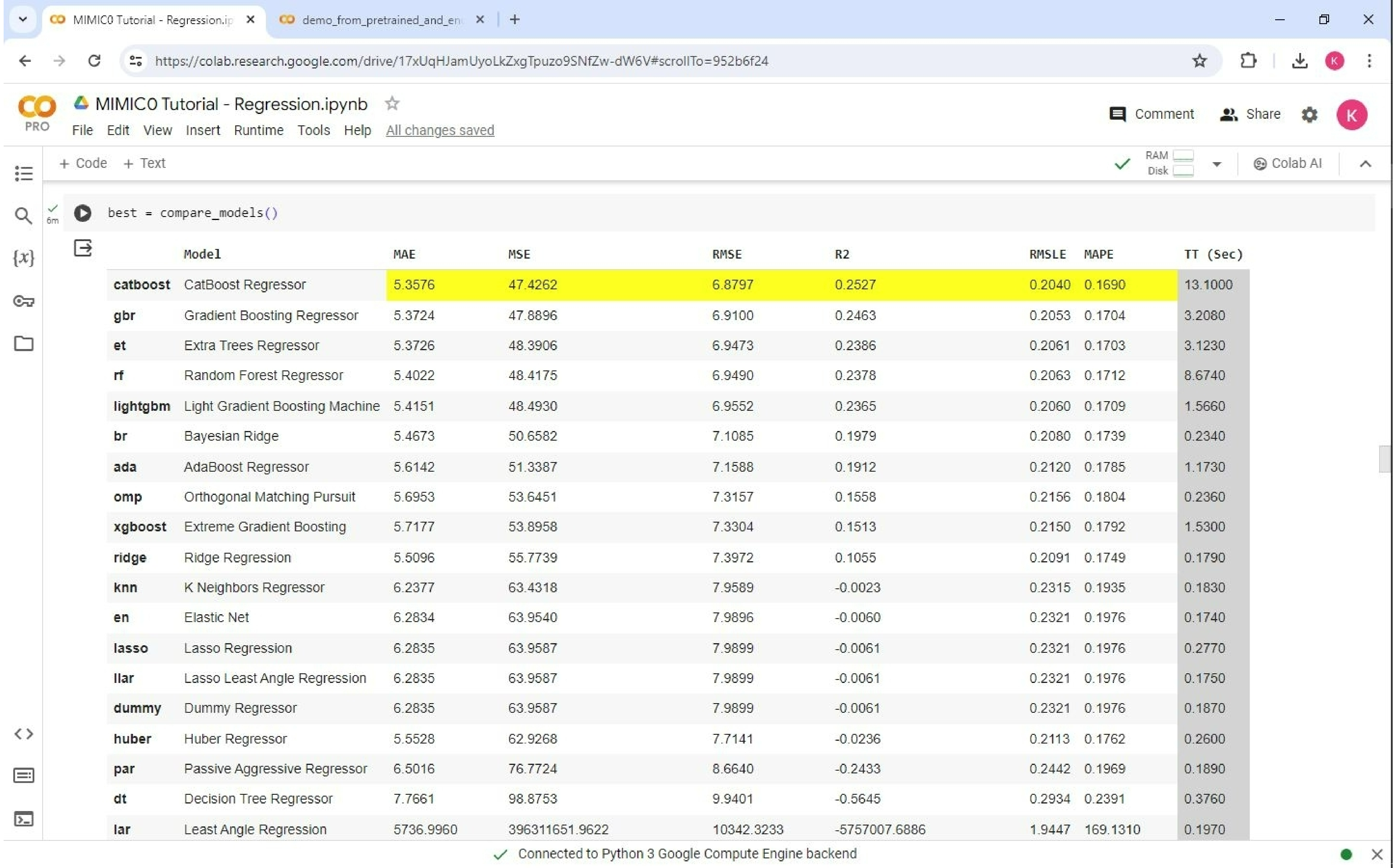
Target Variable Selection

The dataset provides three measures of severity that we use as numeric target variable for regression analysis.

- The Oxford Acute Severity of Illness Score (OASIS)
- The Simplified Acute Physiology Score (SAPS II)
- Sequential Organ Failure Assessment (SOFA) Score https://www.mdcalc.com/calc/691/sequential-organ-failure-assessment-sofa-score https://arxiv.org/abs/2001.10977
- These scores are appropriate for different medical conditions. Due to the shortage of time, I randomly selected OASIS as target.

Mean Absolute Error

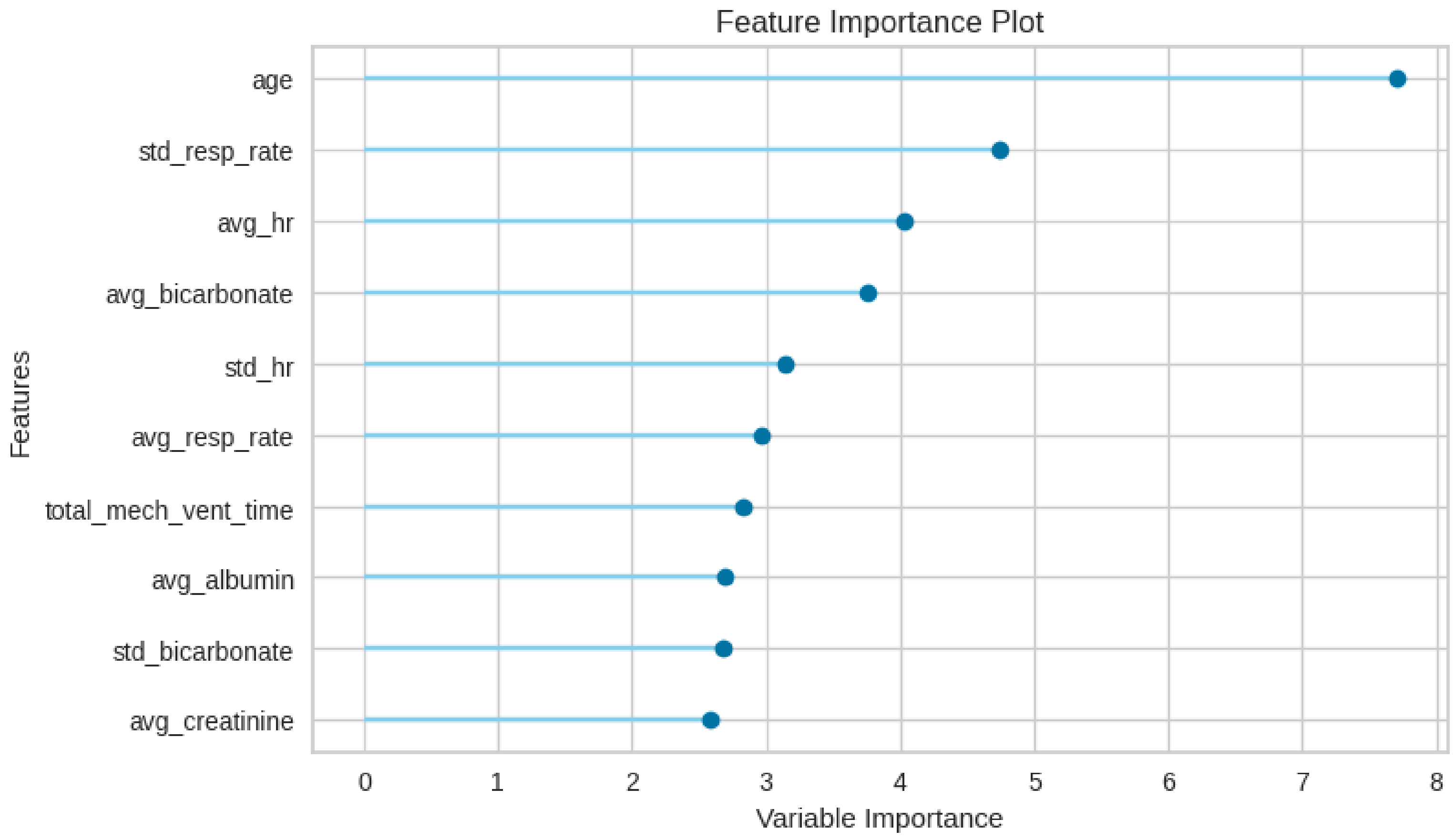
- The next slide shows that the CatBoost Regressor model produces the lowest MAE
- Therefore, It is selected as the model for the regression analysis of this dataset.


MIMIC0 Tutorial - Regression.ip
demo_from_pretrained_and_en
https://colab.research.google.com/drive/17xUqHJamUyoLkZxgTpuzo9SNfZw-dW6V#scrollTo=952b6f24
MIMIC0 Tutorial - Regression.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
All changes saved
Comment
Share
+ Code
+ Text
RAM
Disk
Colab AI
6m

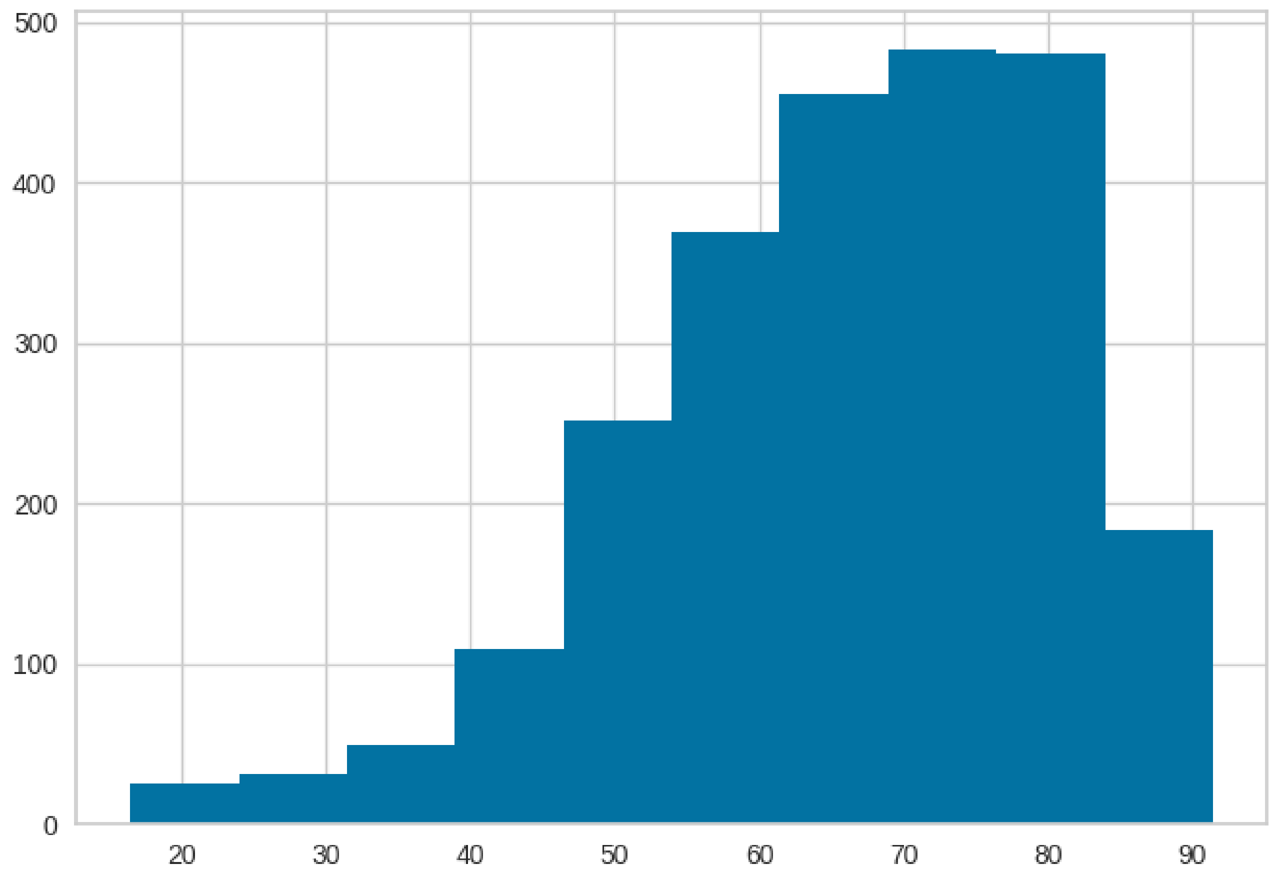
```
best = compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
catboost	CatBoost Regressor	5.3576	47.4262	6.8797	0.2527	0.2040	0.1690	13.1000
gbr	Gradient Boosting Regressor	5.3724	47.8896	6.9100	0.2463	0.2053	0.1704	3.2080
et	Extra Trees Regressor	5.3726	48.3906	6.9473	0.2386	0.2061	0.1703	3.1230
rf	Random Forest Regressor	5.4022	48.4175	6.9490	0.2378	0.2063	0.1712	8.6740
lightgbm	Light Gradient Boosting Machine	5.4151	48.4930	6.9552	0.2365	0.2060	0.1709	1.5660
br	Bayesian Ridge	5.4673	50.6582	7.1085	0.1979	0.2080	0.1739	0.2340
ada	AdaBoost Regressor	5.6142	51.3387	7.1588	0.1912	0.2120	0.1785	1.1730
omp	Orthogonal Matching Pursuit	5.6953	53.6451	7.3157	0.1558	0.2156	0.1804	0.2360
xgboost	Extreme Gradient Boosting	5.7177	53.8958	7.3304	0.1513	0.2150	0.1792	1.5300
ridge	Ridge Regression	5.5096	55.7739	7.3972	0.1055	0.2091	0.1749	0.1790
knn	K Neighbors Regressor	6.2377	63.4318	7.9589	-0.0023	0.2315	0.1935	0.1830
en	Elastic Net	6.2834	63.9540	7.9896	-0.0060	0.2321	0.1976	0.1740
lasso	Lasso Regression	6.2835	63.9587	7.9899	-0.0061	0.2321	0.1976	0.2770
llar	Lasso Least Angle Regression	6.2835	63.9587	7.9899	-0.0061	0.2321	0.1976	0.1750
dummy	Dummy Regressor	6.2835	63.9587	7.9899	-0.0061	0.2321	0.1976	0.1870
huber	Huber Regressor	5.5528	62.9268	7.7141	-0.0236	0.2113	0.1762	0.2600
par	Passive Aggressive Regressor	6.5016	76.7724	8.6640	-0.2433	0.2442	0.1969	0.1890
dt	Decision Tree Regressor	7.7661	98.8753	9.9401	-0.5645	0.2934	0.2391	0.3760
lar	Least Angle Regression	5736.9960	396311651.9622	10342.3233	-5757007.6886	1.9447	169.1310	0.1970

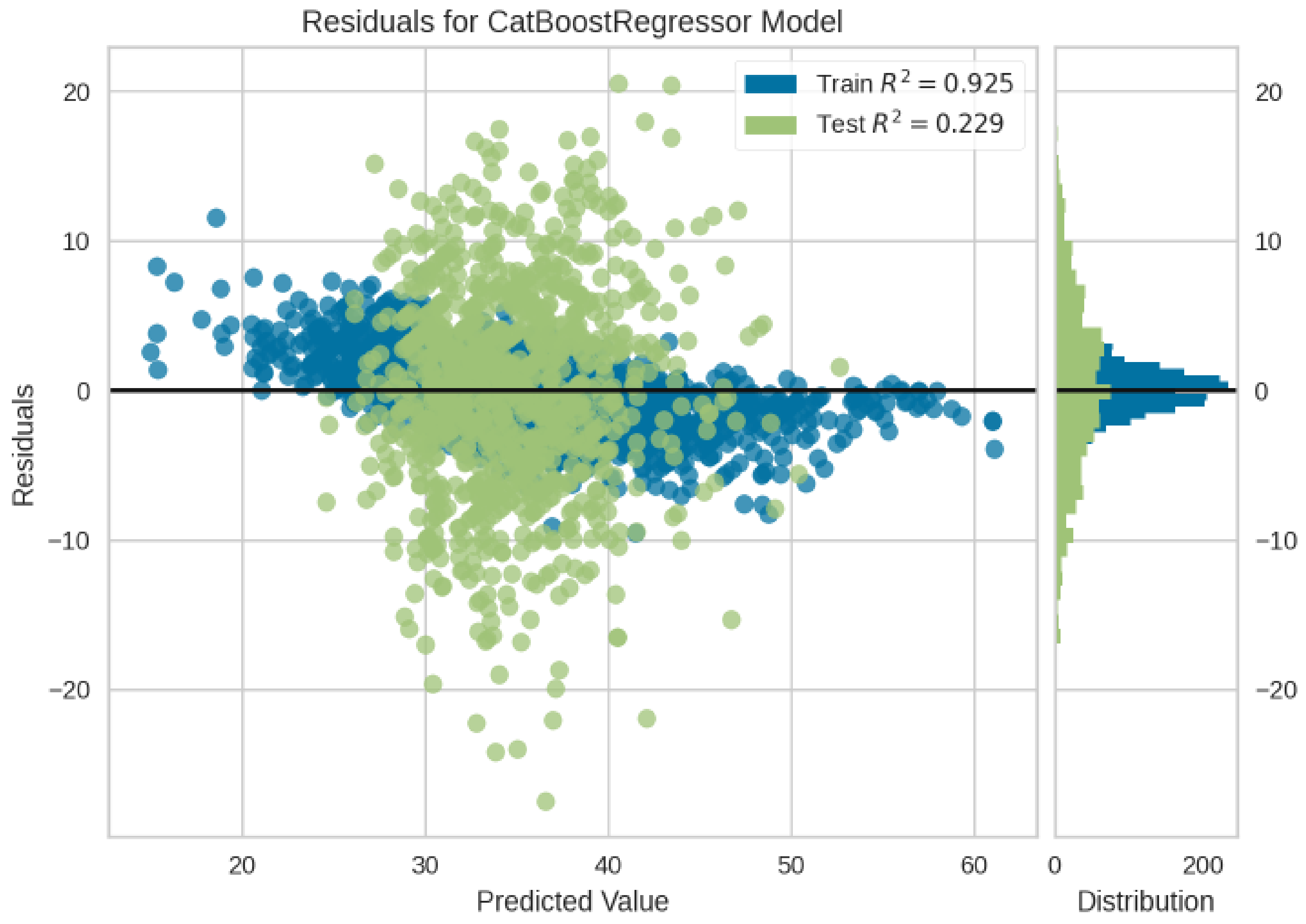
Connected to Python 3 Google Compute Engine backend



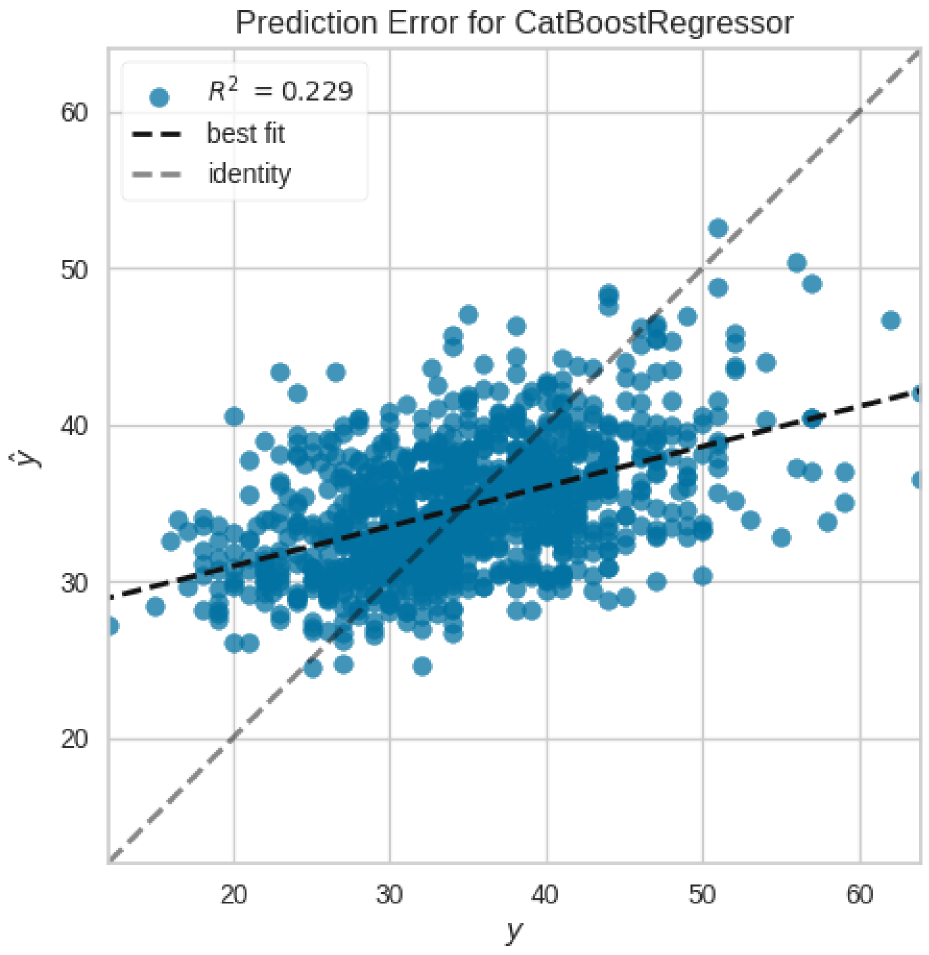
Age is the most important feature for the target variable OASIS severity score



The Age distribution of ICU patients. As expected, the 65 to 85 range is the peak



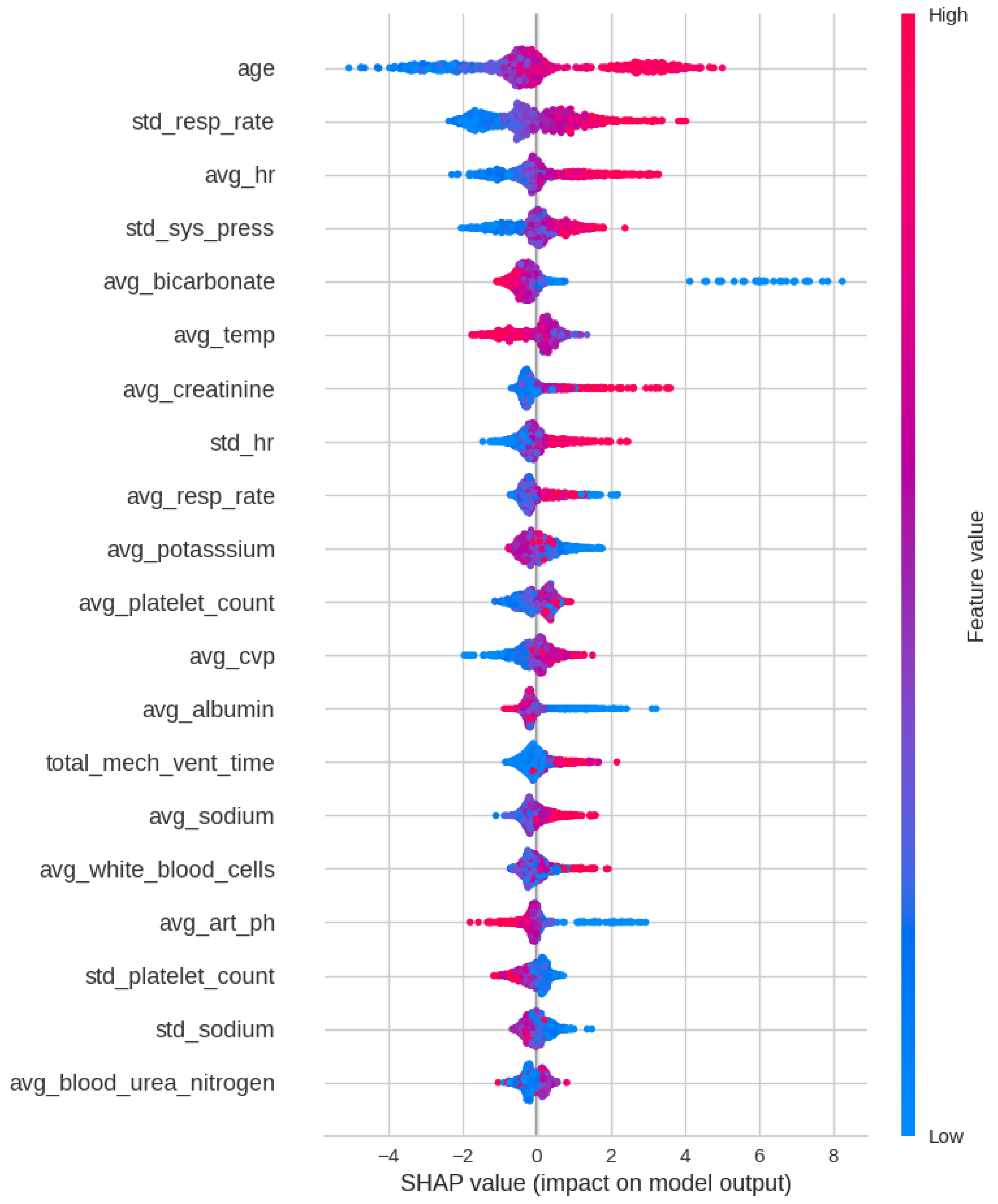
R-square shows how well the model is. The R-square in the above plot show that the model explains the training data well, but prediction of test

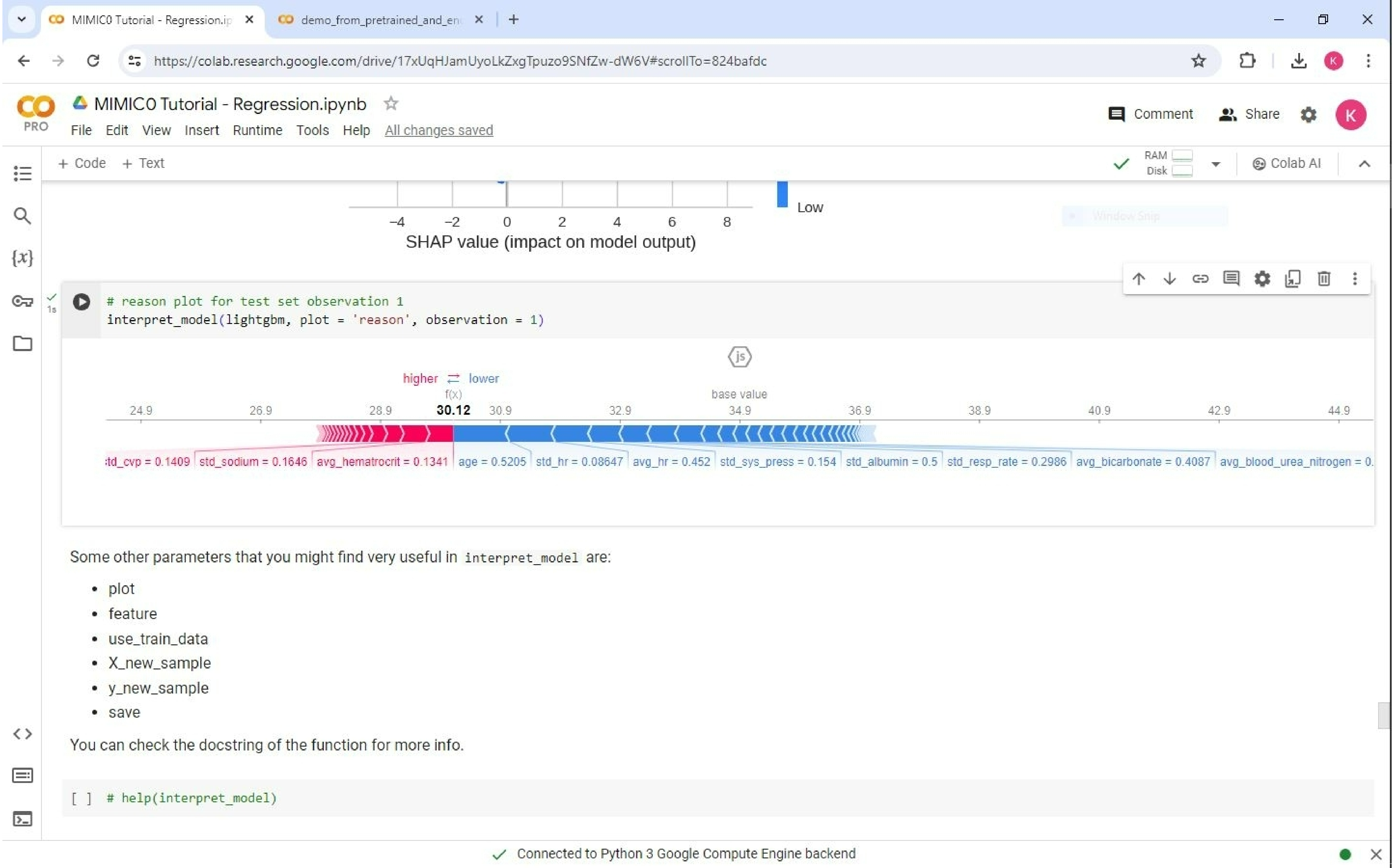


Prediction Error for CatBoostRegressor
R^2 = 0.229
best fit
identity
60
50
40
30
20
20
30
40
50
60
$\hat{y}$
y

SHAP value

- SHAP values are based on game theory and assign an importance value to each feature in a model. Features with positive SHAP values positively impact the prediction, while those with negative values have a negative impact. The magnitude is a measure of how strong the effect is.
- https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability
- The SHAP value in the next two slides shows the importance of the Age factor.


age
std_resp_rate
avg_hr
std_sys_press
avg_bicarbonate
avg_temp
avg_creatinine
std_hr
avg_resp_rate
avg_potasssium
avg_platelet_count
avg_cvp
avg_albumin
total_mech_vent_time
avg_sodium
avg_white_blood_cells
avg_art_ph
std_platelet_count
std_sodium
avg_blood_urea_nitrogen
−4
−2
0
2
4
6
8
SHAP value (impact on model output)
High
Low
Feature value



```
# reason plot for test set observation 1
interpret_model(lightgbm, plot = 'reason', observation = 1)
```

Some other parameters that you might find very useful in `interpret_model` are:

- plot
- feature
- use_train_data
- X_new_sample
- y_new_sample
- save

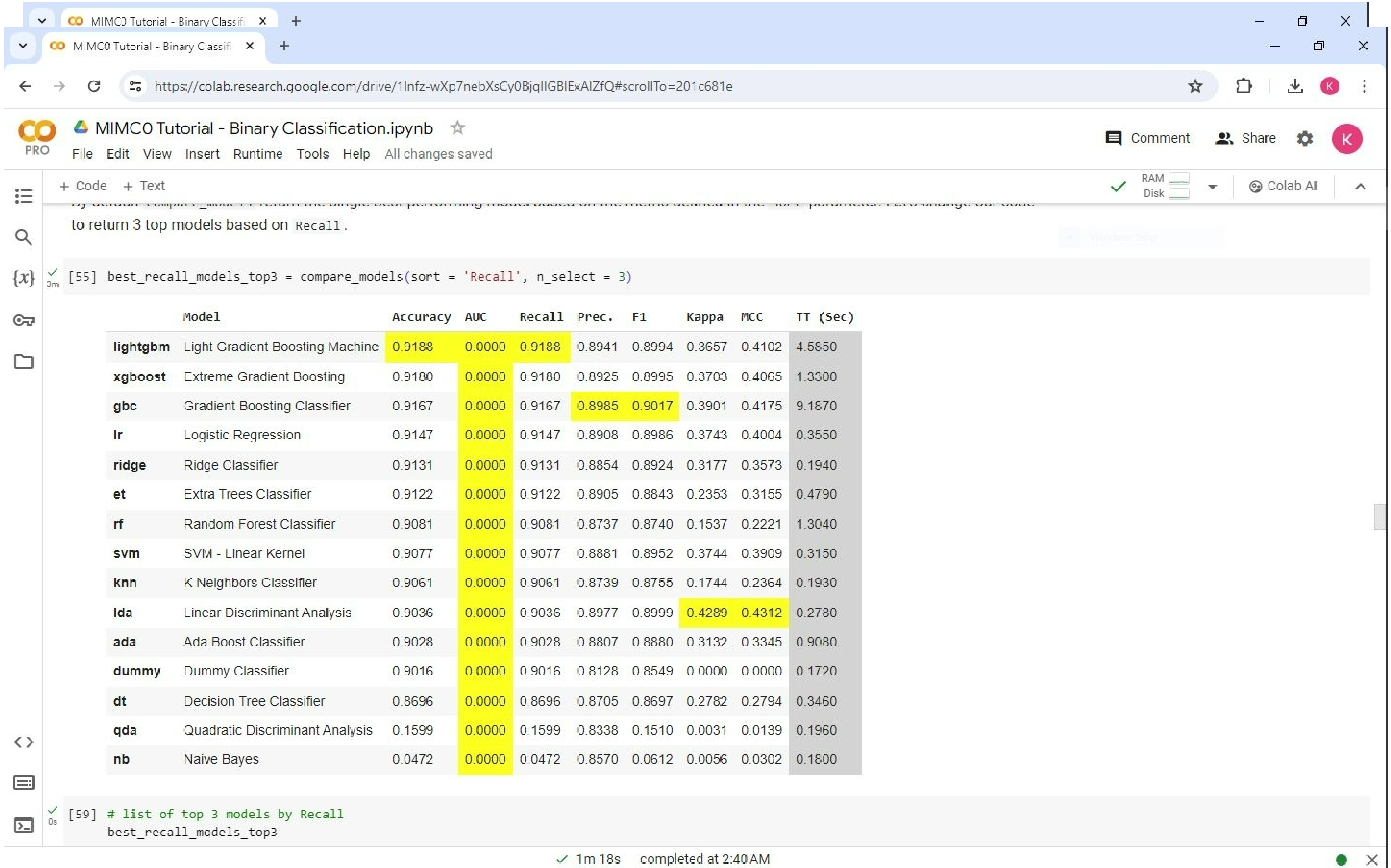
You can check the docstring of the function for more info.

```
[ ] # help(interpret_model)
```

Connected to Python 3 Google Compute Engine backend

Classification Analysis

- https://colab.research.google.com/drive/1Infz-wXp7nebXsCy0BjqlIGBlExAlZfQ?usp=sharing contains the analysis
- There are no many category variables I the dataset. I select SURGERY_FLAG as the target measure, which is a multi-class variable, with the following three possible values, as follows:
- A narrowly defined surgery (**Narrow**) that is usually a major therapeutic procedure
- A more broadly defined surgery (**Broad**) that includes major diagnostic and invasive minor therapeutic procedures
- Neither a narrowly nor broadly defined surgery (**Neither**)
- https://hcup-us.ahrq.gov/toolssoftware/surgeryflags_svcproc/surgeryflagssvc_proc.jsp


MIMC0 Tutorial - Binary Classifi
https://colab.research.google.com/drive/1lnfz-wXp7nebXsCy0BjqIIGBIExAIZfQ#scrollTo=201c681e
MIMC0 Tutorial - Binary Classification.ipynb
PRO
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI

By default compare_models return the single best performing model based on the metric defined in the sort parameter. Let's change our code to return 3 top models based on `Recall`.

```
[55] best_recall_models_top3 = compare_models(sort = 'Recall', n_select = 3)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9188	0.0000	0.9188	0.8941	0.8994	0.3657	0.4102	4.5850
xgboost	Extreme Gradient Boosting	0.9180	0.0000	0.9180	0.8925	0.8995	0.3703	0.4065	1.3300
gbc	Gradient Boosting Classifier	0.9167	0.0000	0.9167	0.8985	0.9017	0.3901	0.4175	9.1870
lr	Logistic Regression	0.9147	0.0000	0.9147	0.8908	0.8986	0.3743	0.4004	0.3550
ridge	Ridge Classifier	0.9131	0.0000	0.9131	0.8854	0.8924	0.3177	0.3573	0.1940
et	Extra Trees Classifier	0.9122	0.0000	0.9122	0.8905	0.8843	0.2353	0.3155	0.4790
rf	Random Forest Classifier	0.9081	0.0000	0.9081	0.8737	0.8740	0.1537	0.2221	1.3040
svm	SVM - Linear Kernel	0.9077	0.0000	0.9077	0.8881	0.8952	0.3744	0.3909	0.3150
knn	K Neighbors Classifier	0.9061	0.0000	0.9061	0.8739	0.8755	0.1744	0.2364	0.1930
lda	Linear Discriminant Analysis	0.9036	0.0000	0.9036	0.8977	0.8999	0.4289	0.4312	0.2780
ada	Ada Boost Classifier	0.9028	0.0000	0.9028	0.8807	0.8880	0.3132	0.3345	0.9080
dummy	Dummy Classifier	0.9016	0.0000	0.9016	0.8128	0.8549	0.0000	0.0000	0.1720
dt	Decision Tree Classifier	0.8696	0.0000	0.8696	0.8705	0.8697	0.2782	0.2794	0.3460
qda	Quadratic Discriminant Analysis	0.1599	0.0000	0.1599	0.8338	0.1510	0.0031	0.0139	0.1960
nb	Naive Bayes	0.0472	0.0000	0.0472	0.8570	0.0612	0.0056	0.0302	0.1800

```
[59] # list of top 3 models by Recall
     best_recall_models_top3
```

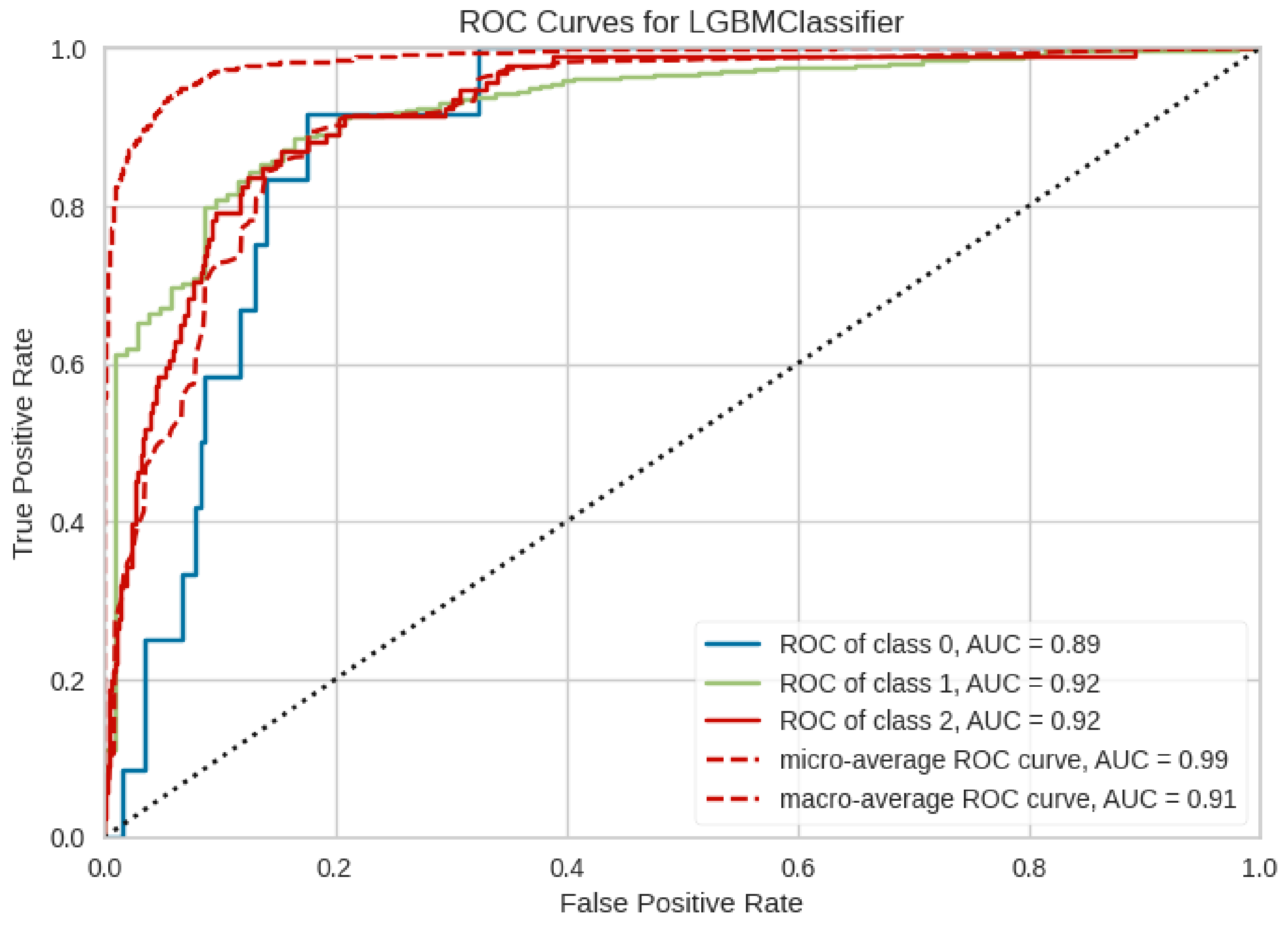
✓ 1m 18s completed at 2:40 AM

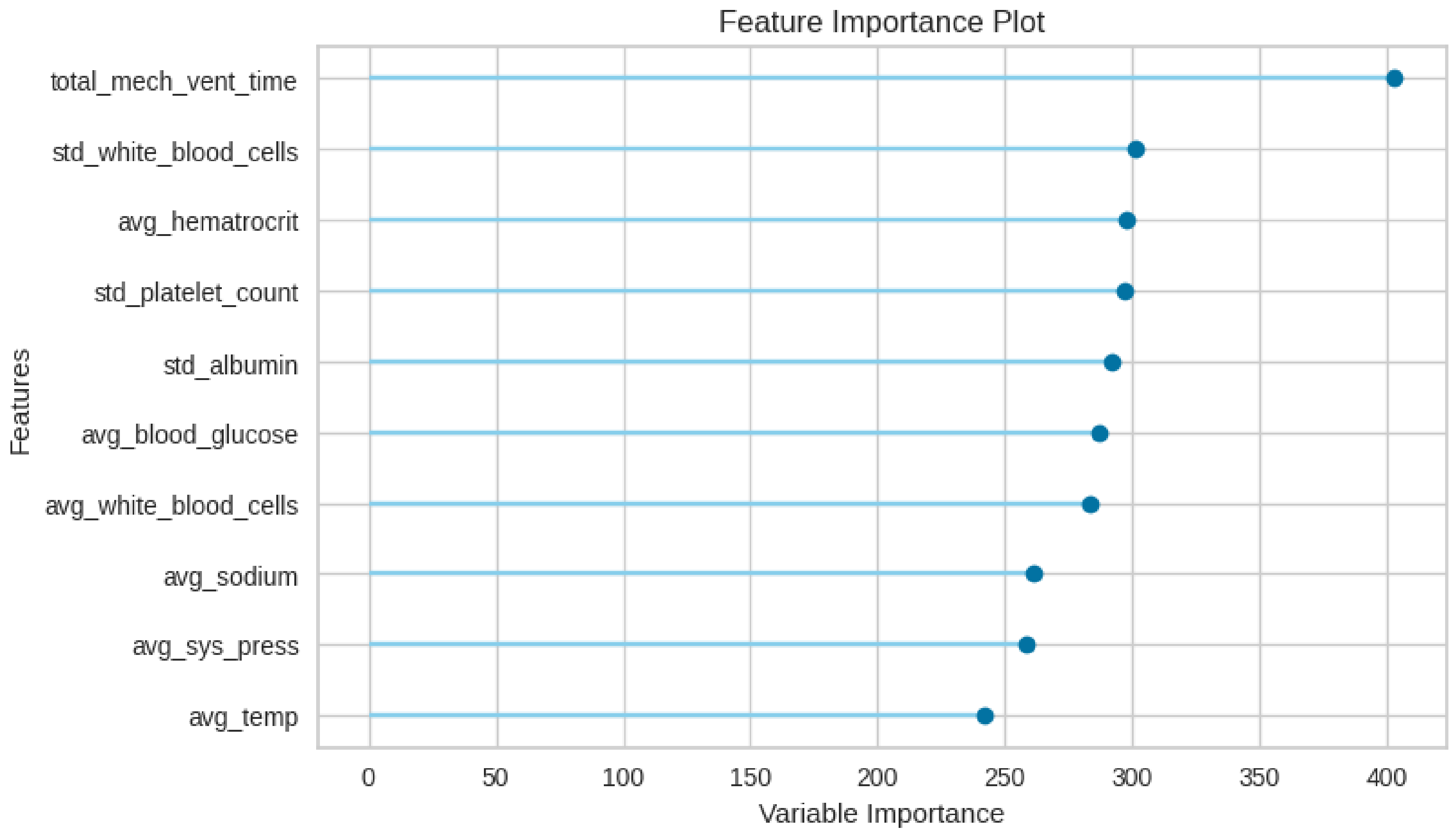
ROC plot

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

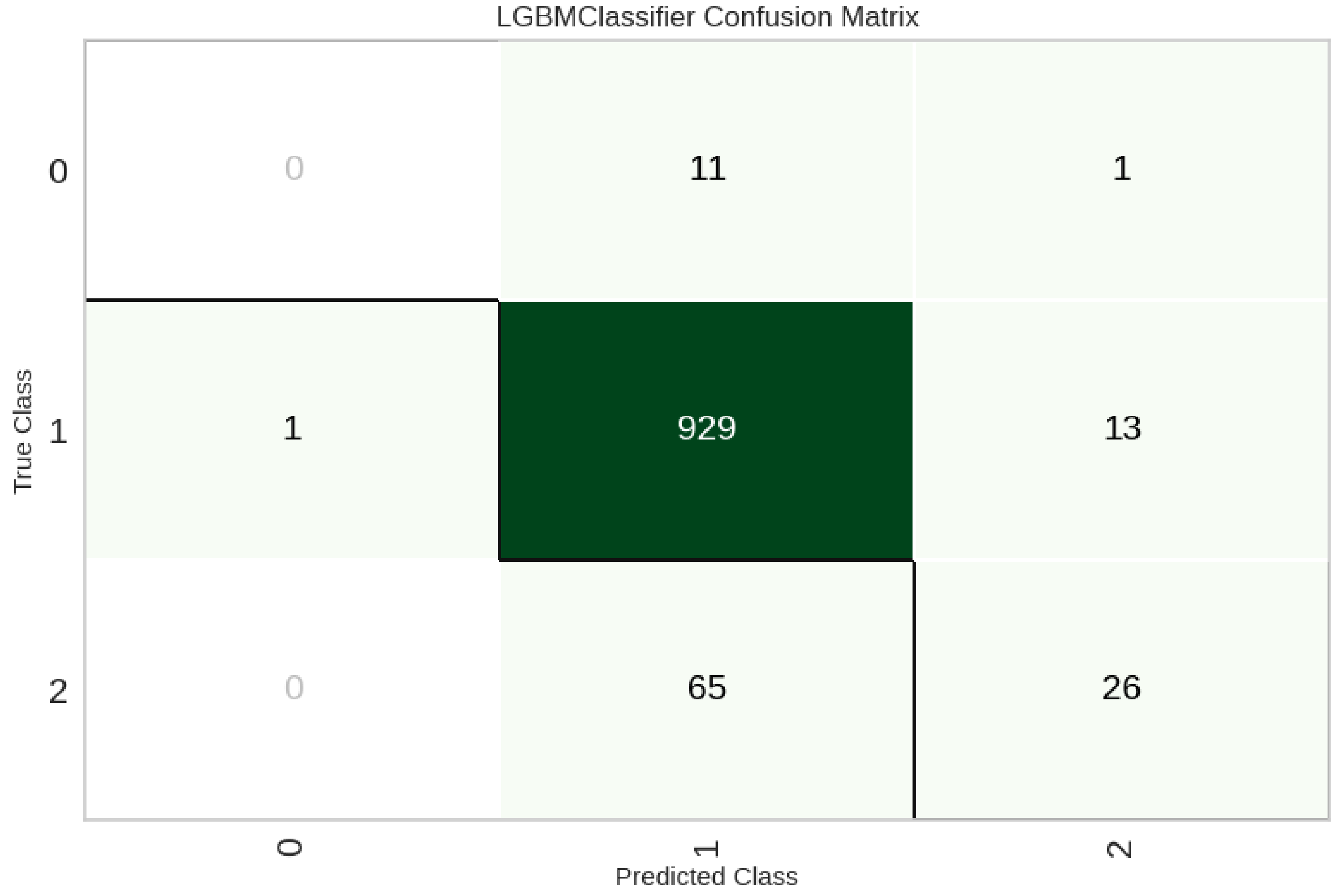

ROC Curves for LGBMClassifier
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC of class 0, AUC = 0.89
ROC of class 1, AUC = 0.92
ROC of class 2, AUC = 0.92
micro-average ROC curve, AUC = 0.99
macro-average ROC curve, AUC = 0.91



Different from the OASIS severity score, mechanical ventilator time, rather than Age, is the top feature for the target variable SURGERY_FLAG.

Confusion Matrix

- A confusion matrix is a table that sums up the performance of a classification model. It works for binary and multi-class classification. The confusion matrix shows the number of correct predictions: true positives (TP) and true negatives (TN).



For a 3-class model, this Confusion matrix with middle square containing highest number and few outliers indicate few bad predictions.

Bonus: Contrastive Learning MedClip

Contrastive learning allows models to extract meaningful representations from unlabeled data. By leveraging similarity and dissimilarity, contrastive learning enables models to map similar instances close together in a latent space while pushing apart those that are dissimilar. This approach has proven to be effective across diverse domains, spanning computer vision, natural language processing (NLP), and reinforcement learning.

Our main reference for the following explanation is

https://encord.com/blog/guide-to-contrastive-learning/

Data Augmentation

Contrastive learning often begins with data augmentation, which involves applying various transformations or perturbations to unlabeled data to create diverse instances or augmented views. The goal of data augmentation is to increase the variability of the data and expose the model to different perspectives of the same instance. Common data augmentation techniques include cropping, flipping, rotation, random crop, and color transformations. By generating diverse instances, contrastive learning ensures that the model learns to capture relevant information regardless of variations in the input data.

Encoder Network

The next step in contrastive learning is training an encoder network. The encoder network takes the augmented instances as input and maps them to a latent representation space, where meaningful features and similarities are captured. The encoder network is typically a deep neural network architecture, such as a convolutional neural network (CNN) for image data or a recurrent neural network (RNN) for sequential data. The network learns to extract and encode high-level representations from the augmented instances, facilitating the discrimination between similar and dissimilar instances in the subsequent steps.

Encoder Network

To further refine the learned representations, a projection network is employed. The projection network takes the output of the encoder network and projects it onto a lower-dimensional space, often referred to as the projection or embedding space. This additional projection step helps enhance the discriminative power of the learned representations. By mapping the representations to a lower-dimensional space, the projection network reduces the complexity and redundancy in the data, facilitating better separation between similar and dissimilar instances.

Projection Network

To further refine the learned representations, a projection network is employed. The projection network takes the output of the encoder network and projects it onto a lower-dimensional space, often referred to as the projection or embedding space. This additional projection step helps enhance the discriminative power of the learned representations. By mapping the representations to a lower-dimensional space, the projection network reduces the complexity and redundancy in the data, facilitating better separation between similar and dissimilar instances.

Contrastive Learning

- The contrastive learning objective is applied once the augmented instances are encoded and projected into the embedding space. The objective is to maximize the agreement between positive pairs (instances from the same sample) and minimize the agreement between negative pairs (instances from different samples). This encourages the model to pull similar instances closer together while pushing dissimilar instances apart. The similarity between instances is typically measured using a distance metric, such as Euclidean distance or cosine similarity. The model is trained to minimize the distance between positive pairs and maximize the distance between negative pairs in the embedding space.

Loss Function

Contrastive learning utilizes a variety of loss functions to establish the objectives of the learning process. These loss functions play a crucial role in guiding the model to capture significant representations and differentiate between similar and dissimilar instances. The selection of the appropriate loss function depends on the specific task requirements and data characteristics. Each loss function aims to facilitate the learning of representations that effectively capture meaningful similarities and differences within the data. In a later section, we will delve into these loss functions in detail.

Training and Optimization

Once the loss function is defined, the model is trained using a large unlabeled dataset. The training process involves iteratively updating the model's parameters to minimize the loss function. Optimization algorithms such as stochastic gradient descent (SGD) or its variants are commonly used to fine-tune the hyperparameters of the model. The training process typically involves batch-wise updates, where a subset of augmented instances is processed at a time. During training, the model learns to capture the relevant features and similarities in the data. The iterative optimization process gradually refines the learned representations, leading to better discrimination and separation between similar and dissimilar instances.

Downstream Task Evaluation: Image Classification vs Object Detection vs Image Segmentation

The ultimate measure of success for contrastive learning is its performance on downstream tasks. The learned representations are utilized as input features for specific tasks such as image classification, object detection, sentiment analysis, or language translation. The model's performance on these tasks is evaluated using appropriate metrics, including accuracy, precision, recall, F1 score, or task-specific evaluation criteria. Higher performance on the downstream tasks indicates better generalization and usefulness of the learned representations.

Transfer Learning

Contrastive learning enables transfer learning, where the presentation of learned representations from one task can be applied to related tasks. Generalization is evaluated by assessing how well the representations transfer to new tasks or datasets. If the learned representations generalize well to unseen data and improve performance on new tasks, it indicates the effectiveness of contrastive learning in capturing meaningful features and similarities.

Comparison with Baselines

To understand the effectiveness of contrastive learning, it is essential to compare the learned representations with baseline models or other state-of-the-art approaches. Comparisons can be made in terms of performance metrics, robustness, transfer learning capabilities, or computational efficiency. Such comparisons provide insights into the added value of contrastive learning and its potential advantages over alternative methods.

Loss Functions in Contrastive Learning

In contrastive learning, various loss functions are employed to define the objectives of the learning process. These loss functions guide the model to capture meaningful representations and discriminate between similar and dissimilar instances. By understanding the different loss functions used in contrastive learning, we can gain insights into how they contribute to the learning process and enhance the model's ability to capture relevant features and similarities within the data

MedClip

- https://github.com/RyanWangZf/MedCLIP
- MedCLIP is a medical image captioning Deep Learning neural network based on the OpenAI CLIP architecture.
- https://github.com/RyanWangZf/MedCLIP provides a simple demo.
- https://colab.research.google.com/drive/11gBL3TQbSwlvHNZXq13Gs6BH4wnvz9EJ?usp=sharing is my attempt to run the demo. Unfortunately, the github archive does not provide a model file.

Summary

- This presentation explains the selection and characteristics of the MIMIC III data set
- We overview and demonstrate the clustering, regression and classification analysis of the dataset.
- We briefly review the results plots generated from these analysis
- These results reveal the important features for selected target variables. We only cover subsets of the 48 columns in the dataset.
- We also overview the use of contrastive learning for analyzing medical imaging.