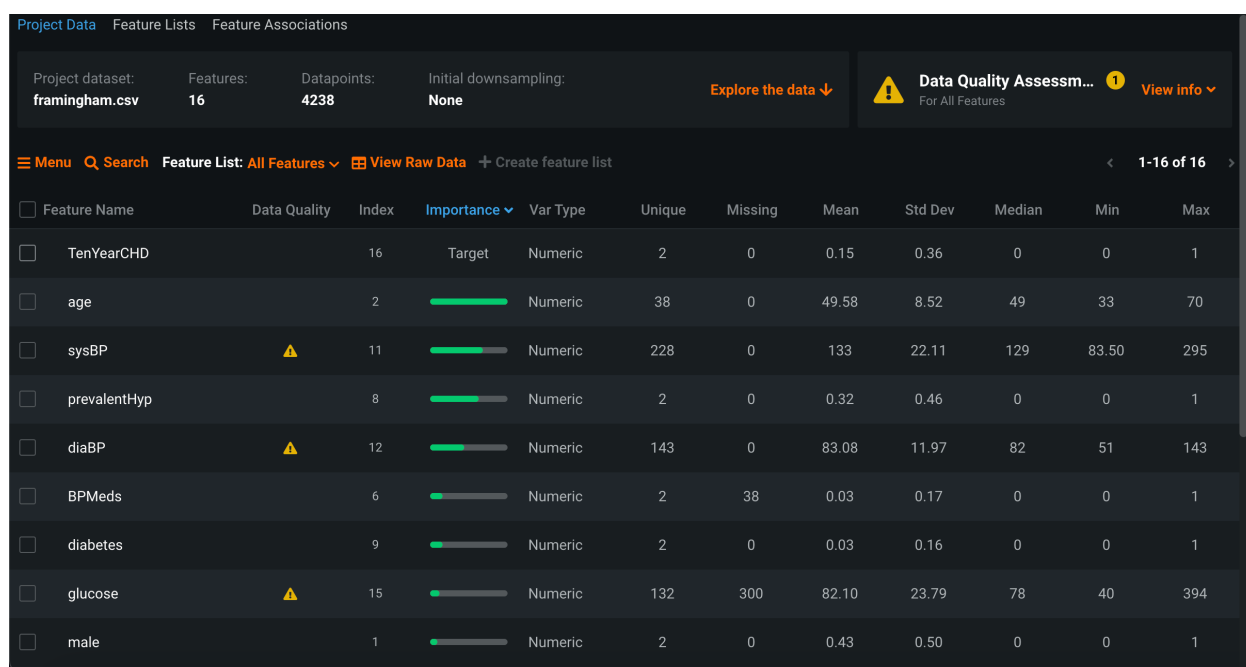


Predictive Modeling with DataRobot

In this assignment, I will use DataRobot to explore Auto Machine Learning and try to tuning the parameter.

First of all, my data comes from Kaggle, which is about heart disease and its target is TenYearCHD. TenYearCHD is a binary data and “1” represents “yes” for 10 year risk of coronary heart disease CHD and “0” represents “no” for 10 year risk of coronary heart disease CHD. The dataset features have sex, age, Current Smoker, BP Meds and so on.




Feature Name	Data Quality	Index	Importance	Var Type	Unique	Missing	Mean	Std Dev	Median	Min	Max
TenYearCHD		16	Target	Numeric	2	0	0.15	0.36	0	0	1
age		2		Numeric	38	0	49.58	8.52	49	33	70
sysBP	⚠	11		Numeric	228	0	133	22.11	129	83.50	295
prevalentHyp		8		Numeric	2	0	0.32	0.46	0	0	1
diaBP	⚠	12		Numeric	143	0	83.08	11.97	82	51	143
BPMeds		6		Numeric	2	38	0.03	0.17	0	0	1
diabetes		9		Numeric	2	0	0.03	0.16	0	0	1
glucose	⚠	15		Numeric	132	300	82.10	23.79	78	40	394
male		1		Numeric	2	0	0.43	0.50	0	0	1

After I uploaded this dataset on DataRobot, it auto analyze the data and show the result to me. We can see that all of the data is numeric and their missing number, min, max and mean. The yellow triangle means that there is an outlier in this feature.

framingham.csv ? 2

Create New Project Manage Projects

framingham.csv    

Total features: 16

Target: TenYearCHD

Metric: LogLoss

Datapoints: 4.24k

Created: 2021-10-30 20:48:49

File Name: framingham.csv

Initial Ingest Downsampling: none

Total # of Models: 63

Positive class: 1

Partitioning Method: Stratified

CV Runs: 5

Holdout Percentage: 20.0095

Model Evaluation: Cross-Validation

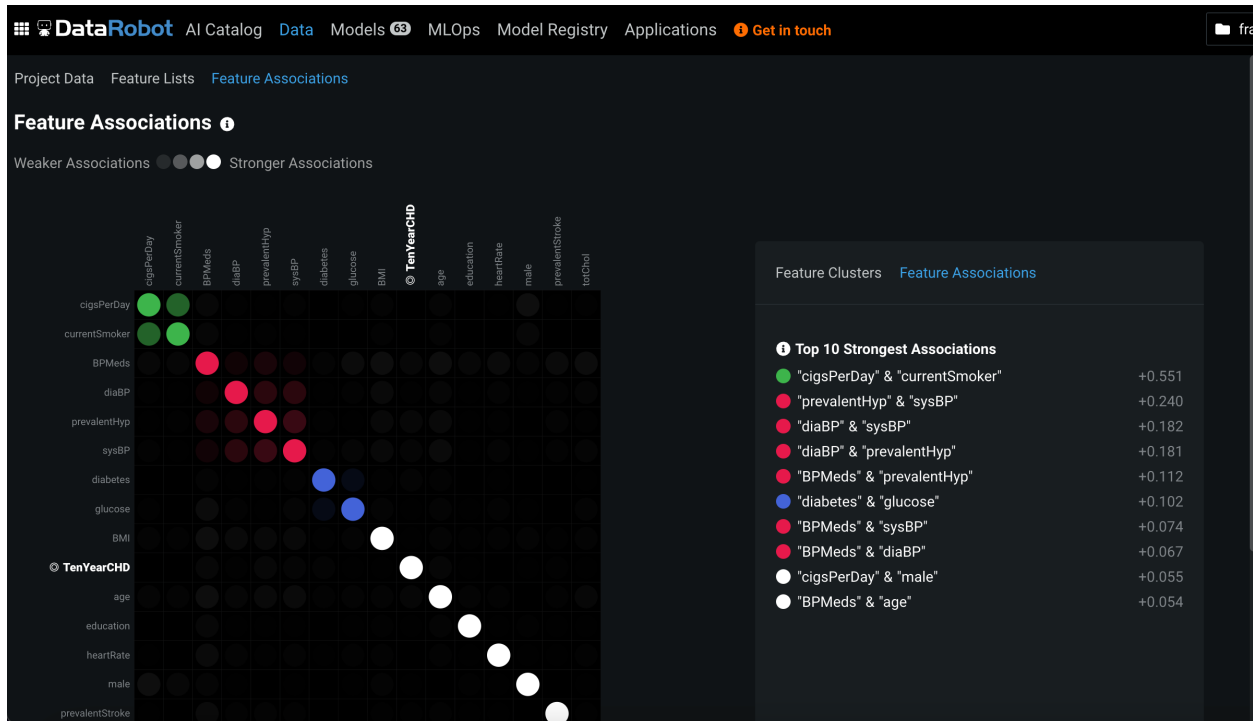
Owner: zwang9@mail.yu.edu

Show less ^

Using 0 of 8 total workers across all projects

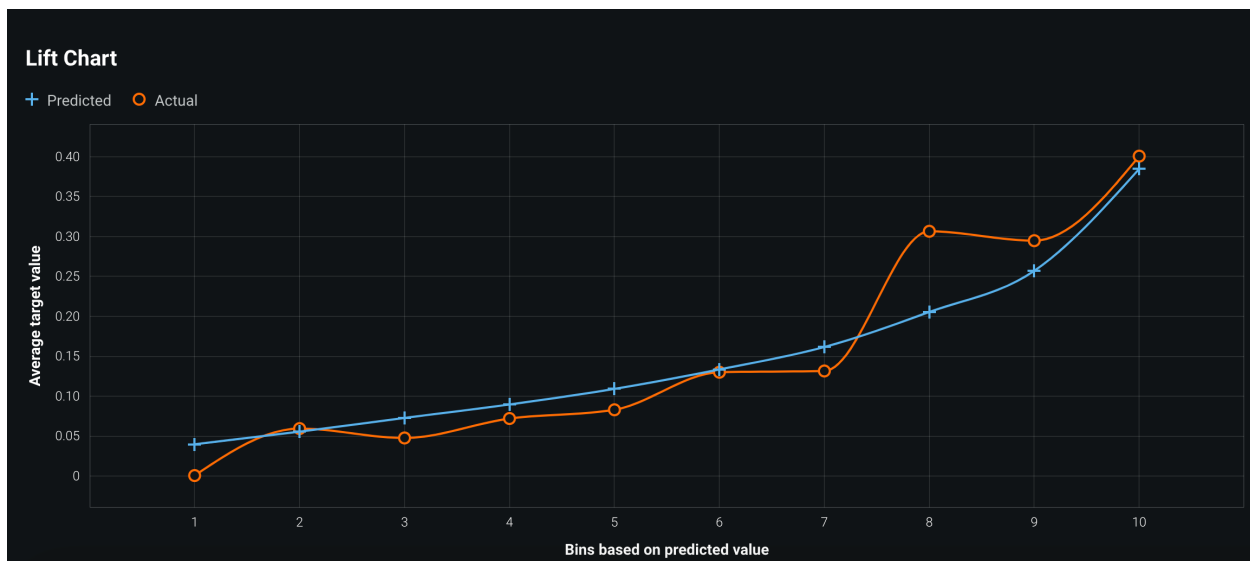
When we click on the right data-name button, we can see the Hyperparameter of this dataset. From this screenshot, we can know there are 63 models run on this dataset. The holdout data percentage is 20.0095%. Why is it the float number? I think maybe it used the number of datasets to times 20% and got a float number for the holdout data. But

the number must be int. So it rounds the result and uses the approximate number of holdouts to divide by the total number of dataset and finally got such a float number 20.0095. We also can know the data size and the number of features from this screenshot.

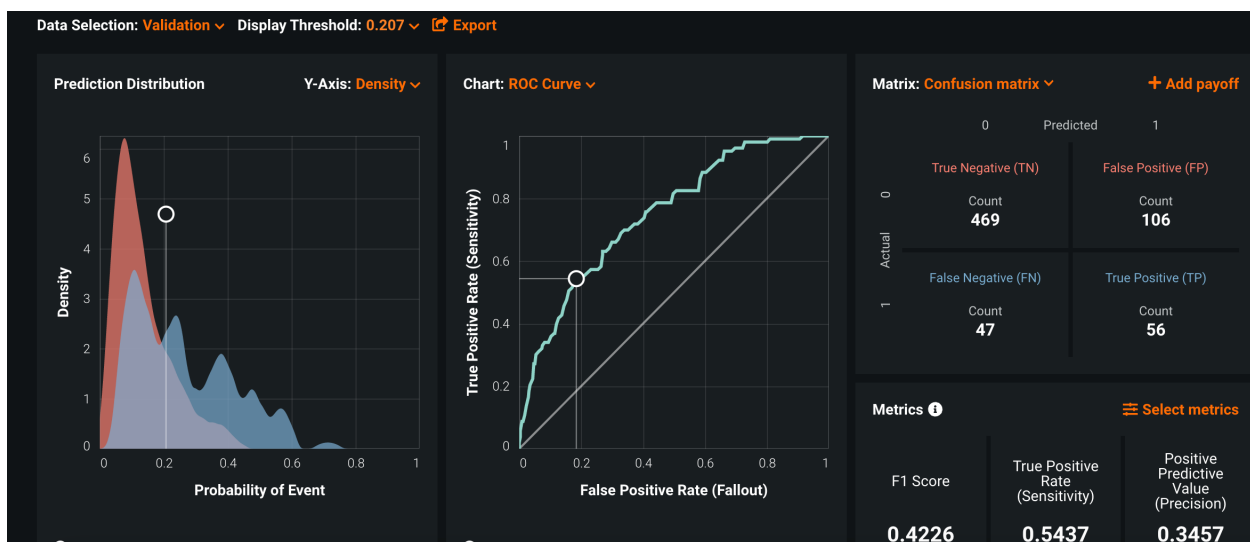


When I clicked on the feature association in the Data tag, I can see the heat plot and know the relationship between each feature. Obviously, "cigsPerDay" and "currentSmoker" have the highest correspond. If we want to calculate the PCA, we can eliminate one of their features.

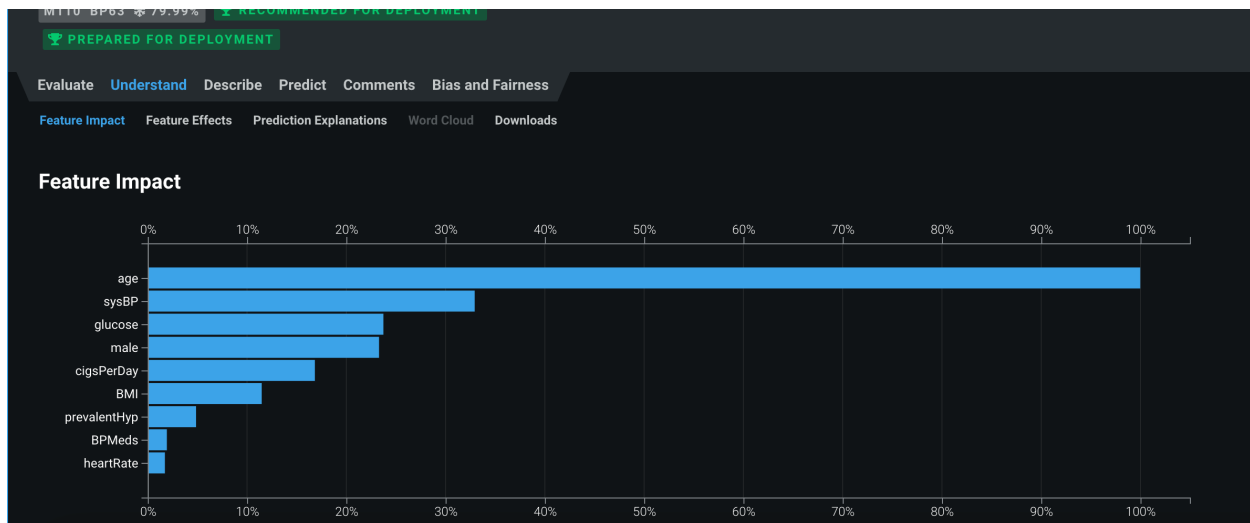
Repository Leaderboard Insights Learning Curves Speed vs Accuracy Model Comparison						WORKERS	
Menu Search + Add new model Filter Models Deploy Automodel Export						Using 0 of 8 total workers across all projects 08	
Model Name & Description						STATUS	
Feature List & Sample Size						Autopilot has finished	
Validation						ACTIONS	
Cross Validation						Get More Accuracy	
Holdout						Next Modeling Mode: Comprehensive	
						Feature List: Informative Features	
						Configure modeling settings	
						Unlock project Holdout for all models	



Here we can see the Lift Chart. The predicted is very close to the Actual.



When we choose ROC Curve, we can see that the roc curve is far away from the diagonal and close to the right button. It means that is model is useful. And DataRobot also helps me to show the confusion matrix and calculate its F1, sensitivity and precision.



This is one of the functions that I like about the datarobot tool. Although it is easy to calculate the coefficient of the dataset, we can easily know which feature has the biggest influence on the predicted target.

☐ Model Name & Description Feature List & Sample Size Validation **Cross Validation** Holdout

gamma
0.002711251127

max_sample
None

n_components
1000

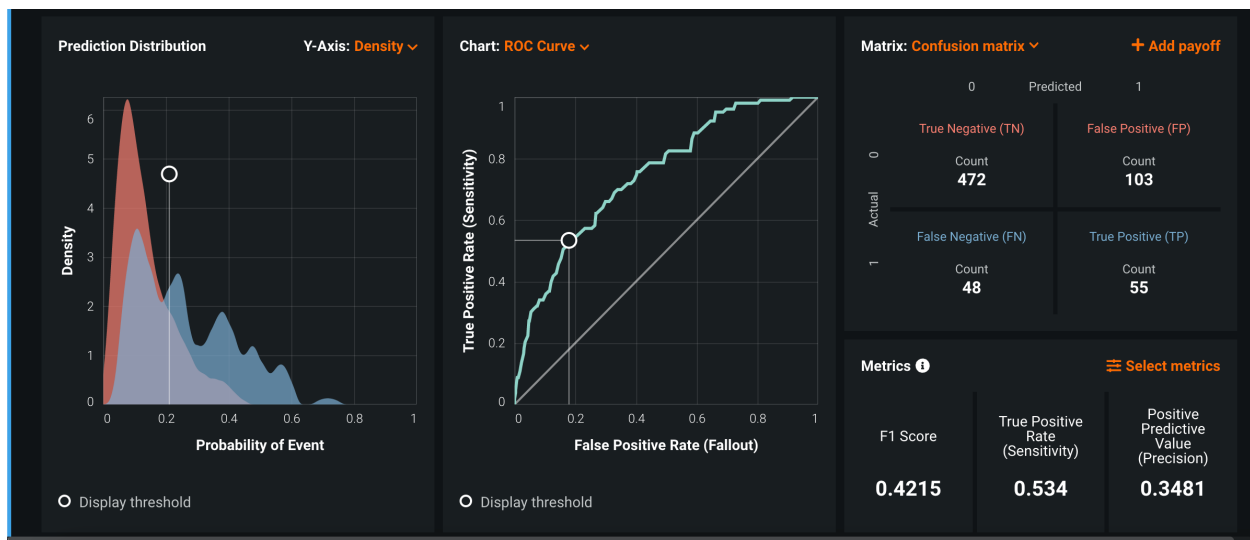
random_state
23

smart_sampling
True

subsample
1.0

tol
0.0001

I tried to adjust the data on the tuning page and changed the `n_components` from 500 to 1000 and change its `random_state` from 1234 to 23.



Its count of true negative was improved and got a better precision.

Model Name & Description: 2.7825594022071245

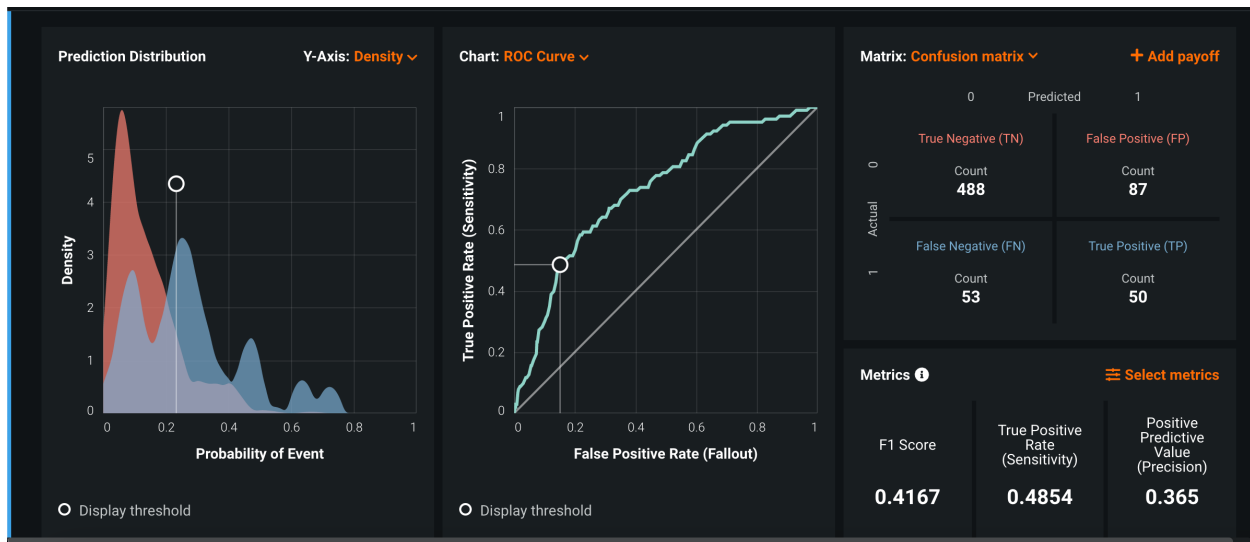
Feature List & Sample Size:

Validation: [Cross Validation](#) | Holdout

Model Parameters:

- gamma: 0.1
- max_sample: None
- n_components: 500
- random_state: 1234
- smart_sampling: True
- subsample: 1.0
- tol: 0.00001

I try to large its gama and reduce its tol.



Also, we got a higher count of true negative and higher precision.

These two try means that we got a lower AUC and didn't get a better model.

The form displays the following parameters:

- approx:** fourier (highlighted with a red box)
- c:** 2.7825594022071245
- gamma:** 0.002711251127
- max_sample:** None
- n_components:** 500
- random_state:** 1234
- smart_sampling:** True
- subsample:**

I tried again to change its approx from balanced_nystroem to fourier.

Menu Search + Add new model Filter Models Deploy Automodel Export		Metric AUC	
Model Name & Description	Feature List & Sample Size	Validation	Cross Validation
Nystroem Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Prediction API Enabled M110 BP63 79.99% RECOMMENDED FOR DEPLOYMENT PREPARED FOR DEPLOYMENT	DR Reduced Features M55 100.0 %	0.7493 *	0.7194 *
Fourier Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with approx=fourier M130 BP63 TUNED	DR Reduced Features M55 100.0 %	0.7488 *	0.7201 *

We can see that we improve the AUC in cross-validation.

approx

nystroem

c

2.7825594022071245

gamma

0.002711251127

max_sample

None

n_components

500

random_state

1234

smart_sampling

True

I tried another approx.

Regularized Linear Model Preprocessing v20 Prediction API Enabled M110 BP63 79.99% RECOMMENDED FOR DEPLOYMENT PREPARED FOR DEPLOYMENT	DR Reduced Features M55 100.0 %	0.7493 *	0.7194 *
Fourier Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with approx=fourier M130 BP63 TUNED	DR Reduced Features M55 100.0 %	0.7488 *	0.7201 *
Nystroem Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with n_components=1000, random_state=23 M127 BP63 TUNED	DR Reduced Features M55 100.0 %	0.7492 *	0.7194 *
Nystroem Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with approx=nystroem M132 BP63 TUNED	DR Reduced Features M55 100.0 %	0.7494 *	0.7193 *

And we got the worst model. I found the sort tool of DataRobot website really can help us to easily test the tuning.

The screenshot shows the DataRobot tuning interface with the following parameters:

- n_components**: 500
- random_state**: 1234
- smart_sampling**: False (highlighted with a red box)
- subsample**: 1.0
- tol**: 0.0001

Below the parameters, there is a "Search type" dropdown set to "Select search option" and a "Describe this tuning" text area. At the bottom right, there is a "Begin Tuning" button.

I tried to close the smart sample and see what would happen.

Menu	Search	+ Add new model	Filter Models	Deploy Automodel	Export	Metric AUC
<input type="checkbox"/>	Model Name & Description	Feature List & Sample Size	Validation	Cross Validation		
Hydrogen Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Prediction API Enabled						
DR Reduced Features M55 100.0 %						0.7493 * 0.7194 *
M110 BP63 * 79.99% RECOMMENDED FOR DEPLOYMENT						
PREPARED FOR DEPLOYMENT						
Fourier Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with approx=fourier						
DR Reduced Features M55 100.0 %						0.7488 * 0.7201 *
M130 BP63 TUNED						
Nystroem Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with smart_sampling=False						
DR Reduced Features M55 100.0 %						0.7493 * 0.7194 *
M133 BP63 TUNED						
Nystroem Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with n_components=1000, random_state=23						
DR Reduced Features M55 100.0 %						0.7492 * 0.7194 *
M127 BP63 TUNED						

We can see that the close smart sample also didn't let me get a better model. I thought why I don't try to combine these several "good" changes together and see what would happen.

Menu
Search
Add new model
Filter Models
Deploy Automodel
Export
Metric AUC

☐ Model Name & Description
Feature List & Sample Size
Validation
Cross Validation
Holdout

fourier
C
2.7825594022071245
gamma
0.002711251127
max_sample
None
n_components
1000
random_state
23
smart_sampling
False
subsample
1.0

Menu
Search
Add new model
Filter Models
Deploy Automodel
Export
Metric AUC

☐ Model Name & Description
Feature List & Sample Size
Validation
Cross Validation

Nystroem Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with n_components=1000, random_state=23 M127 BP63 TUNED	DR Reduced Features M55 100.0 %	0.7492 *	0.7194 *
Nystroem Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with approx=nystroem M132 BP63 TUNED	DR Reduced Features M55 100.0 %	0.7494 *	0.7193 *
<input type="checkbox"/> Fourier Kernel SVM Classifier Regularized Linear Model Preprocessing v20 Tuned from M110 with approx=fourier, n_components=1000, random_state=23, smart_sampling=False M134 BP63 TUNED ☆	DR Reduced Features M55 100.0 %	0.7472 *	0.7188 *

And I got a nearly worst model. It seems like that DataRobot has already helped us to tuning the parameter and help us to choose the best model. Near all of the parameters were tested by DataRobot and it's a lower probability to find a better model than the model choice by DataRobot.

This is really a good website to use to choose a model. Compared with python, this website can auto help us to load many suitable models and train them. If we use python to build a model, we just can test several models which keep in our mind. What's more, DataRobot also help us to

do much thing such as plot the ROC, calculate F1, AUC, confusion matrix, heat plot and so on. The most important is that we nearly can't do so much thinking in such a little time. Even when we try to adjust some parameters on the model, we may need to spend lots of time to find the variable in python, but DataRobot can easily help us to quickly change it and retrain the model. Furthermore, in python, we may often forget to split data into training data and test data. DataRobot auto helps us to quickly split them into training data and holdout data. And it also helps us to do cross-validation. I think the convenience and quickness of DataRobot are what I am most concerned about.

By the way, I don't want to put python in the opposite of DataRobot. Maybe we can combine them together and let we develop better. DataRobot is a super well website to choose and build a model, but sometimes python is more flexible. Maybe we can use DataRobot to help us choose the model and use python to do something else. For example, when we design a wholly new model and we want to compare it with another existing model. We can use DataRobot to run the data in the existing model and use python to build our new model then compare them. Both of them are very well tool.