User Interface Design

for

Machine Learning App

Prepared by: Eric Chen

First started:

05/07/2025

Last updated:

11/11/2025

1. Introduction

In machine learning, we generate predictive models using certain data and then evaluate the performance of those models using metrics like the ROC curve, precision-recall curve, the confusion matrix, etc. When we do that, we need to find a way to store those models and their performance results in a way that is easy to find

and then easy to visualize them in a user-friendly way. In other words, we need an efficient way to store ML models and their performance results in an organized manner such that we can easily retrieve them at any time. As a result, we created a web application designed to provide the user with an easy-to-use toolkit to develop and test machine learning models, analyze the performance of those models, and then store them in an organized manner. The goal of this is for the user to do those tasks without having to do all the complicated programming in order to accomplish them.

The purpose of this document is to provide the user, business, and developer with an understanding of the approach taken to design the user interface for the whole application. The scope of this document will cover the complete user experience and interface design, detailing what each user page would look like, as well as the functionality of each page.

2. System Overview

The Machine Learning App is a web application designed to provide the user with an easy-to-use toolkit to develop and test machine learning model(s) and to analyze the performance of machine learning model(s). This aims for the user to do those tasks without having to do all the complicated programming needed to accomplish these tasks. This web application is also designed to organize machine learning models and their results in a way that is easy for the user to retrieve and then visualize. It does so by storing the models and their performance results in an organized and well-structured database.

The app has three main sections and two minor sections. The three main sections are the training section, the testing section, and the visualization section. Each selection will have multiple options depending on the type of model that is developed and tested.

The **training section** will allow the user to choose a varying set of configuration options to develop a machine learning experiment. (This can range from what preprocessing steps it should take, what algorithms to use, the hyperparameter feature space to use, etc.) A machine learning experiment in this scenario is a set of machine learning models with varying algorithms and configurations that are trained (an/or tested) on a singular dataset on a specified set of input and output variables.

The **testing section** allows the user to access the developed models in a completed experiment from the database, upload a compatible dataset to test on, choose a set of varying configurations, test the model on that testing set, and finally store the results in the database.

The **visualization section** allows the user to access the performance results of model(s) from the database and look at things like the ROC curve, precision-recall curve, confusion matrix, etc. This page also allows the user to compare performances on different models with an option of different filters.

The three minor sections are the management section, the database list section, and the data preprocessing and engineering section. The **management section** allows the user to delete either a specific ML experiment result, the entire ML experiment itself, or a stored dataset. This section allows the user to rename a ML experiment or a stored dataset. The **database list section** gives a list of all of the completed ML experiments and stored datasets, along with their metadata or basic information. The **data preprocessing section** allows the user to upload a data set and perform data processing and engineering steps and then save it into the database before doing any model training.

So far, this ML App only works with supervised binary classification problems with tabular data where the outcome labels are either 0 or 1. In the future, we will implement binary problems with labels other than 0 or 1 as well as multi-class classification problems into this app.

3. Design Considerations

Assumptions and Dependencies:

Assumptions:

- The user has a basic understanding of machine learning and the basic steps of the machine learning process.
- The user knows certain metrics (like the ROC Curve, confusion matrices, etc.) that are important for analyzing the performance of a machine learning model.
- The user has the data sets available to perform tasks in the machine learning app.
- The user has all of the necessary libraries like Numpy, Pandas, Scikit-learn, etc installed with the correct versions.

Dependencies:

- The app required a connection to a MongoDB database.
- The user has to have the data sets available to perform machine learning tasks.

- Libraries like streamlit, sklearn, numpy, matplotlib, pandas, Autogluon, and others must be installed and some may have a specific version required in order to make this app work.

General Constraints:

- MongoDB rejects certain object types like Sklearn models, AutoGluon models, Pandas dataframe, Int64 objects, and Numpy arrays.
- Some versions of certain libraries may not be compatible with the rest of the app.
- Some models may take a long time to train and test.
- The data sets uploaded to be used for developing and testing a model must be either an Excel or a CSV file.
- The data set must have the correct column set.

Goals and Guidelines:

Goal:

- Provide a user with an easy-to-use interface to perform the tasks in the machine learning process, from training, to testing to analyzing performance.
- Be able to train and test multiple models of varying configurations on a singular dataset.
- To organize the machine learning experiments and their results more conveniently.
- To compare the performance of multiple machine learning models in a convenient way.

Guidelines:

- The user must have the training and testing sets with matching feature sets, label column(s), and file type.
- The configuration the user chooses must be compatible with datasets and the type of machine learning model used. For example, feature selection cannot choose more features than the number of features the raw data sets has.
- Some specific configurations of training a machine learning model do not work for certain ML algorithms or model types.

Development Methods:

- We plan to develop and test each part of the application individually.

- We then piece all these parts together to get the whole web application functioning.
- We receive feedback from people who tested this app and use them to make improvements.
- We tested this app on different environments, such as testing on different computers and with varying datasets.

4. System Architecture

4.1 Database (Backend)

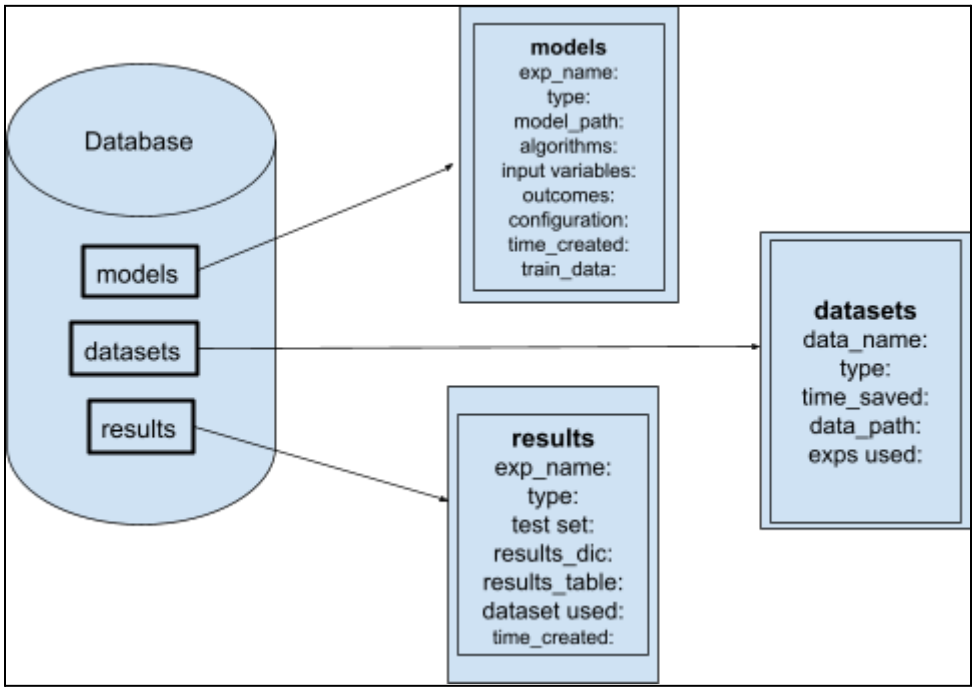


Figure 1.1

This web application uses MongoDB to create and use a database (shown in **Figure 1.1**) to store trained models and performance data. It has three components, *models*, *results,* and *datasets*. The *models* collection of the database stores the name of the experiment (a set of ML models for a set of algorithms/outcomes for a specific dataset), the type of the model, the file path to the machine learning model dictionary itself in the folder system, the algorithms used, the input variables used, the output variables (outcomes) used, the configuration, the time created, and the name of the dataset used for training. The *results* collection stores the name of the experiment (same as the exp. name for the associated model(s)), type of model used, the name of the test, a dictionary of the performance metrics as well as the predictions, probabilities, and ground truths, a table consisting of the performance metric scores, like AU-ROC, accuracy, precision, recall, etc, the name of the dataset used for testing, and the time

it was created. The *datasets* collection stores a dataset that is automatically saved in the database after the user uploaded it in either the training or testing section and begins training or testing. The user can later retrieve that saved dataset for a future ML training or testing experiment. This collection stores the name of the dataset, the type it is (either Train or Test set), the time it was saved, the file path it is located in the folder system, and the list of experiment it is used on so far.

All three collections with all of their contents and metadata are also stored in folders in a file system.

4.2 Main Page

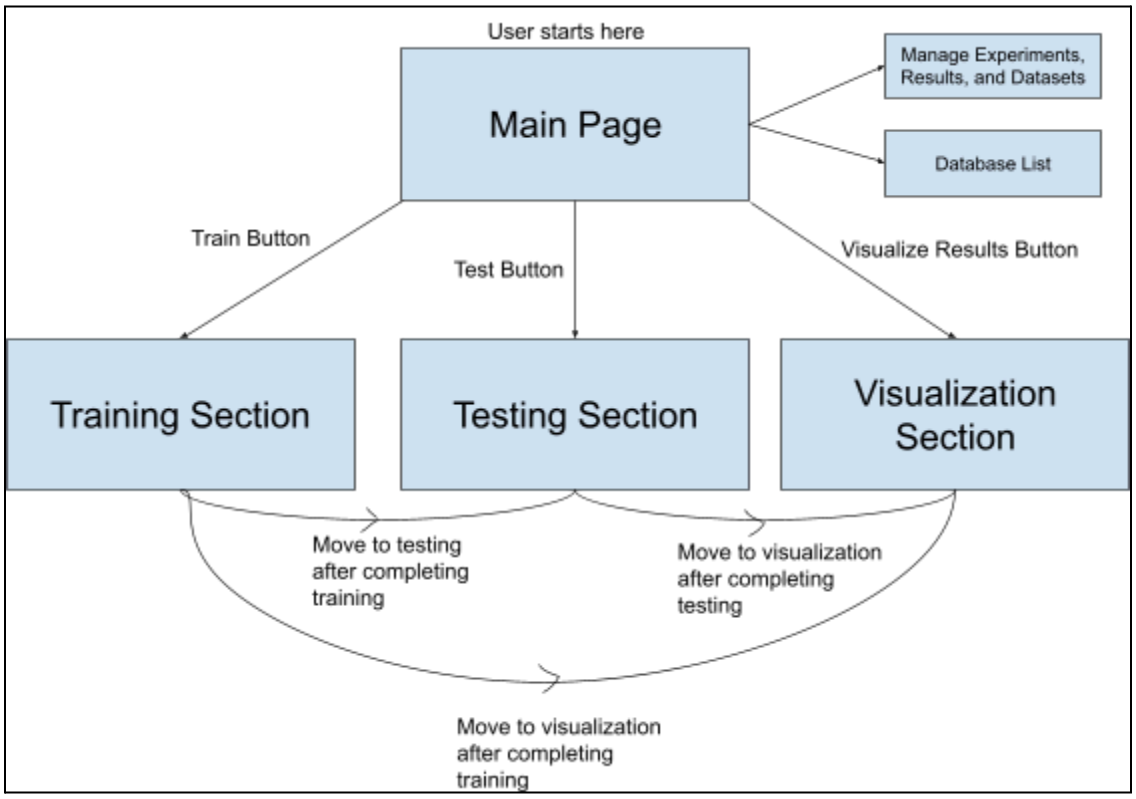


Figure 1.2

Figure 1.2 shows the basic architectural structure of the front end for the machine learning webapp. When the user launches this application, they will start on the Main Page where they will set a total of 5 options each representing a specific section of the app as described earlier. The user from there can go to any of these five different sections of the app. Some sections might be mapped to multiple pages where the user can go depending on the type of machine learning model being used. For instance, the training section can have one page for training with sklearn models, and another for Autogulon models. The testing section can have a page for testing Autogulon models

and another for testing Sklearn models. For the training sections and testing sections, after the user completes the respective task there, they can move directly to the next section that does the next phase of the machine learning process. In other words, the user can go directly from the training stage, to the testing stage, and to the visualization stage. They can also jump to the visualization section directly from the training sections if the option is available (it is if the user does a train/test or cross validation experiment) All three sections have access to the backend of the app, that is, the MongoDB database.

4.3 Training Section

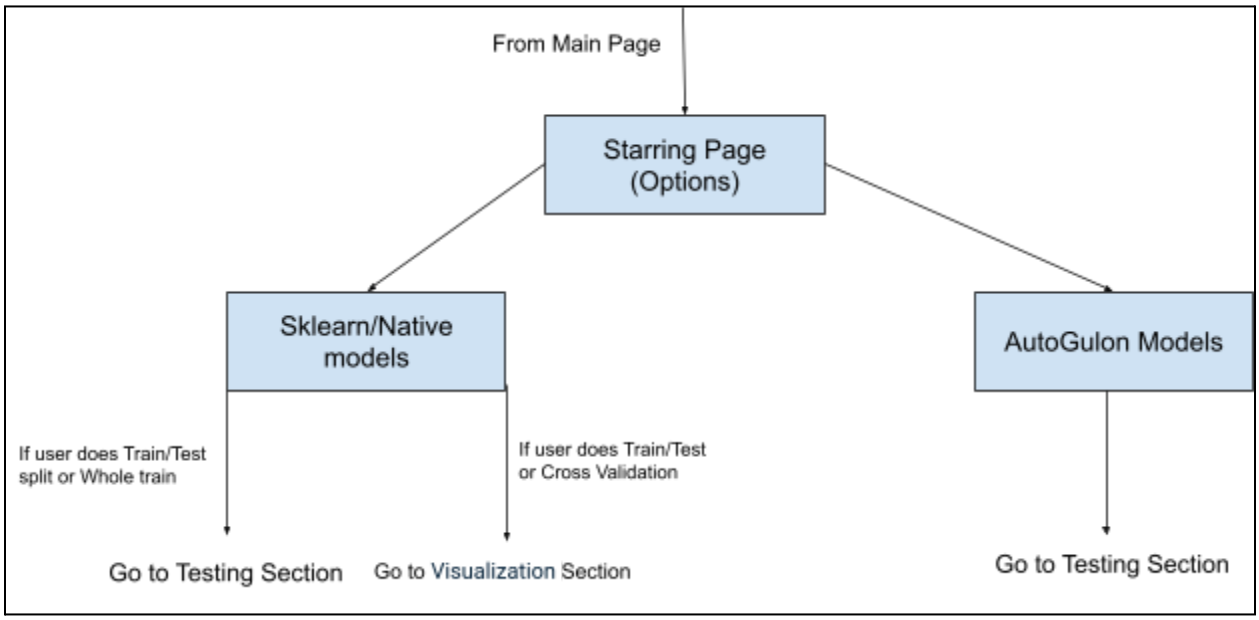


Figure 1.3

Figure 1.3 shows the page map within the training section for the machine learning webapp. The user will be able to choose which type of model they want to train with. It can be either the traditional Sklearn/native models, or the more advanced AutoGulon models.

For the sklearn models, the user can choose 3 options on how they can develop models. The first is training the whole dataset, meaning the user uses the whole entire dataset that's uploaded to train ML models on. No testing or validation step is done with this option. The user then will have to move to the testing section and upload a new compatible testing set to test the new models on. The second option does a traditional train-test split with a user-defined test set size on the uploaded training set. The model is trained on the training section of the data and then tested on the remaining dataset

aside for testing. The user can then either go directly to the visualization section to analyze the test results that were just generated or go to the testing section to test the new models on new data. The third option is the basic cross-validation method. This is only for getting model results and visualization, and should not be used for model generation/creation nor separate experiment testing. Users can also go directly to the visualization section after completing the cross-validation experiment. Users should not be able to go directly to the testing section after cross validation is complete since this option does not produce a set of models that can be tested on in the future.

The AutoGulon section of the training phase has the user use the AutoGulon framework to develop a machine learning model. Much of the AutoGulon framework automates many steps in the machine learning process, which makes it easier for the user to set up a machine learning experiment and/or model development. As a result, there is only one way for users to train with AutoGulon which is similar to the “training the whole dataset” option from the Sklearn section. After training is complete, user can go directly to the AutoGulon testing section to test the newly generated models on new data.

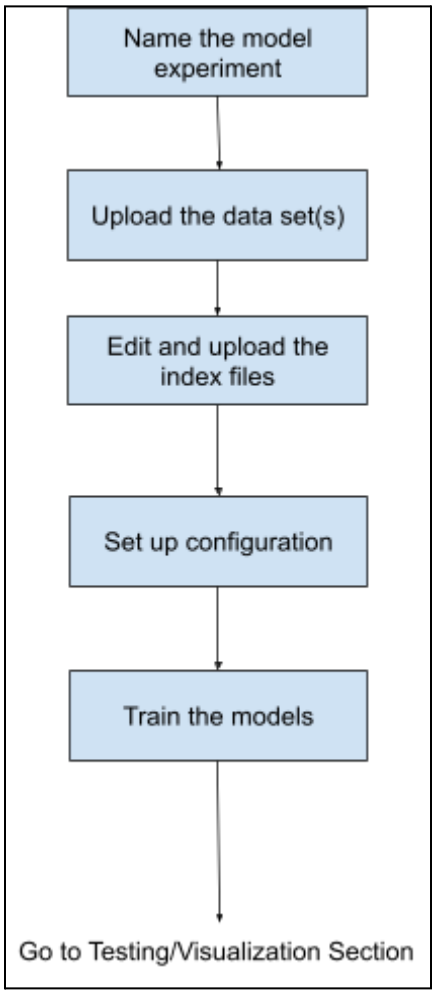


Figure 1.4

Figure 1.4 shows the basic functional flow of a training page. All versions of the training page will go like this: Name the model experiment, upload a data set for training, download an index file whose template is generated from the uploaded training set, customize the index file to indicate which columns of the data are the input or output variables, either generate/upload a customized configuration file or customize the configuration from the interface, and finally train the models. After the training, the page will show the user the button to redirect them to either the testing or visualization page. This whole training pipeline should produce a ML experiment that is a set of models stored in a named experiment generated from the training phase. It also contains metadata including what input variables it used, what output variables/outcomes it is trying to predict, the name of the training set it is used on, etc. Please look back on the models collection in the database section (4.1) on this page to see more detail.

4.4 Testing Section

The testing section of this app has two different testing options for either Sklearn/Native models or AutoGulon models. However, the flow of either testing scenario is the same.

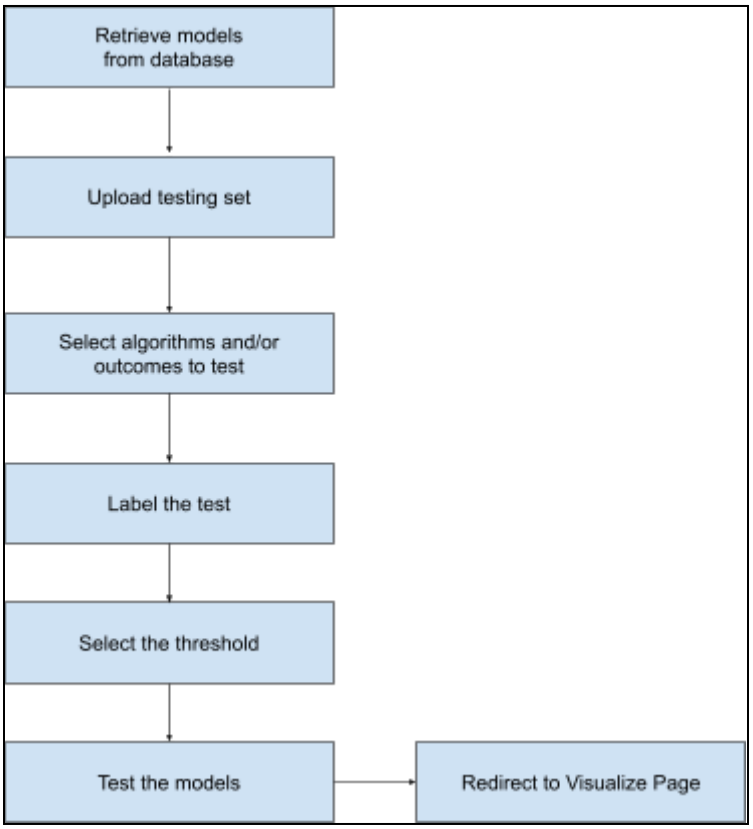


Figure 1.5

Figure 1.5 shows the basic flow of a testing page. First, the user must select a ML experiment from the database. The ML experiment is a set of models stored in a named experiment generated from the training phase. It also contains metadata including what input variables it used, what output variables/outcomes it is trying to predict, the name of the training set it is used on, etc. Then the user must either upload the testing set or retrieve a saved one from the database. The testing set must contain all the features and labels that were involved in the experiment . Then the user can choose any set of algorithms and/or outcomes/labels that are available in this experiment . The user then must name the testing experiment of this model so that we can differentiate between multiple tests from the same experiment. The user next selects the threshold used to decide the classification during testing. After all of this, the user then clicks on the “Test the models” button to perform the actual testing. Once testing is complete, the user can click on the “Visualize ML Results” button at the bottom of the page to redirect to the visualization section.

4.5 Visualization

The visualization section, like the testing section, has multiple pages for each type of model (Sklearn/Native, AutoGulon, Cross Validation). Each page though goes through virtually the same flow.

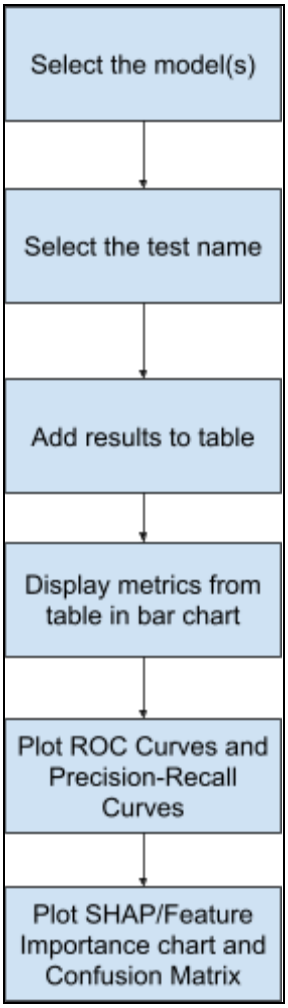


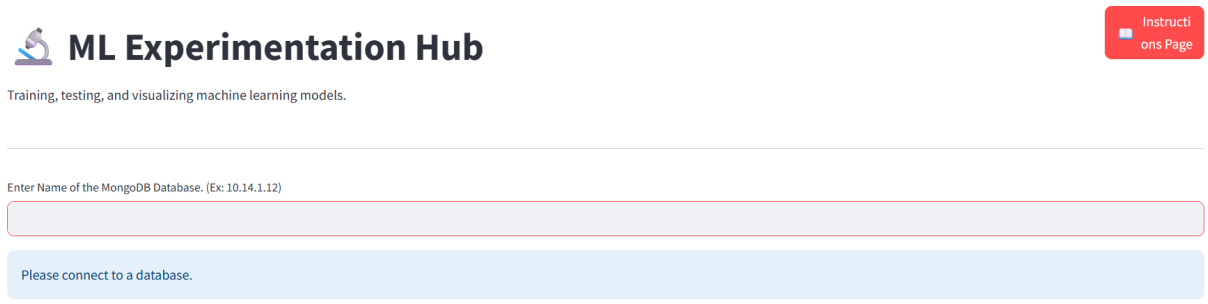
Figure 1.6

Each page in the visualization section should start with the user selecting the model set and the name of the test saved in the database. Then the user adds those results to the table. The user can add as many sets of test results to the table as possible as well remove any one of them. The user can look at the bar chart below the table, select the metric, and filter by algorithm or outcome to compare the numbers on each machine learning model performance metric. The user can then plot as many of the ROC curves from any of the model results uploaded to this page and compare them. The user can also see the Precision-Recall Curve and SHAP/Feature Importance charts below the ROC section. They can also see the confusion matrix below the SHAP/Feature Importance section. The user has the option to clear all plots and tables.

5. Detailed System Design

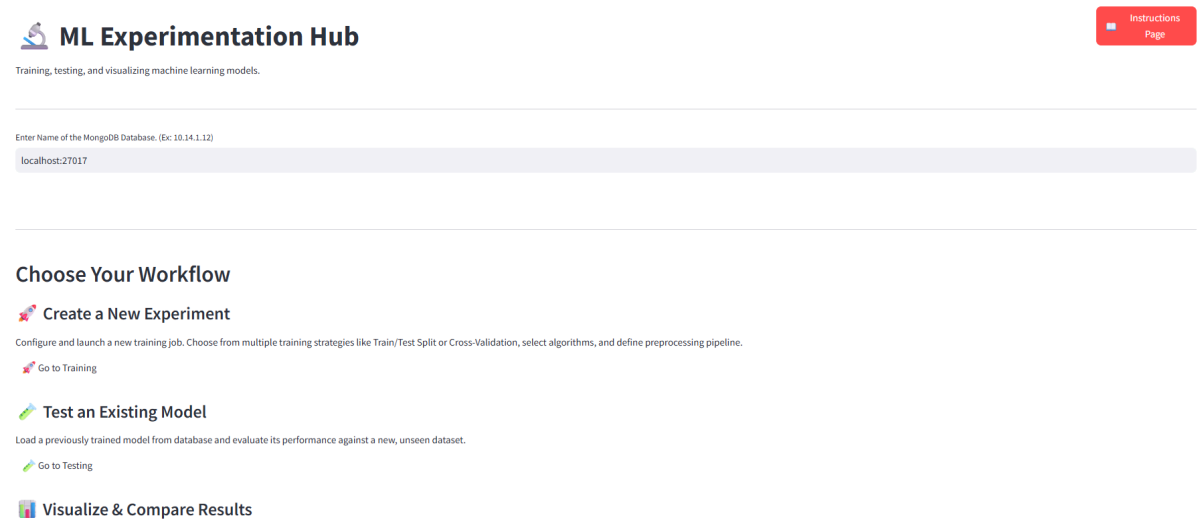
5.1 Main Page

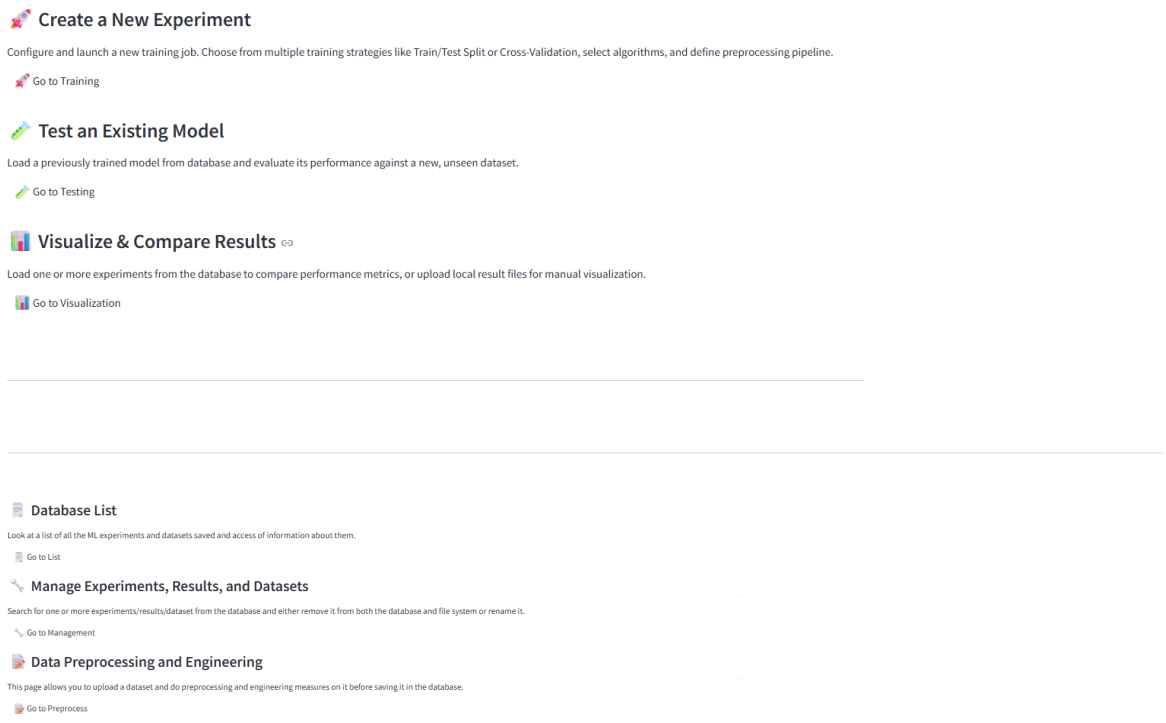
Screenshot of the main page when the user first opens the app



When the user first opens the ML App, the screen above is what he/she will see. There is a title named “ML Experimentation Hub”, a red button in the top right corner that will lead you to the instructions section, and then you have a textbox where you will have to enter the MongoDB connection name to connect to a database.

Screenshot of the main page after the user connects to a MongoDB database

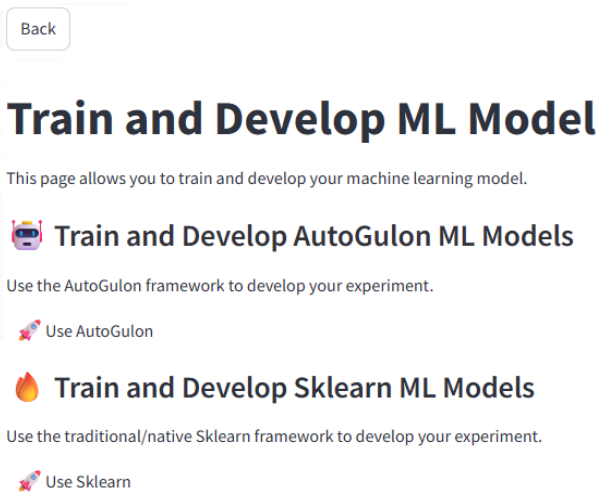




After the user enters the connection name, they will see a set of options leading them to different sections of the app, done so by a button. The top three options have buttons that will lead to the training, testing, and visualization sections of the app. The bottom three lead to the database list section (where the user can see all the experiments and datasets stored in the database along with their metadata), the management section of the app (where the user can delete or rename experiments, tests, and datasets in the database), and the data preprocessing section (where the user can do data preprocessing steps on an uploaded dataset to be saved into the database).

5.2 Training

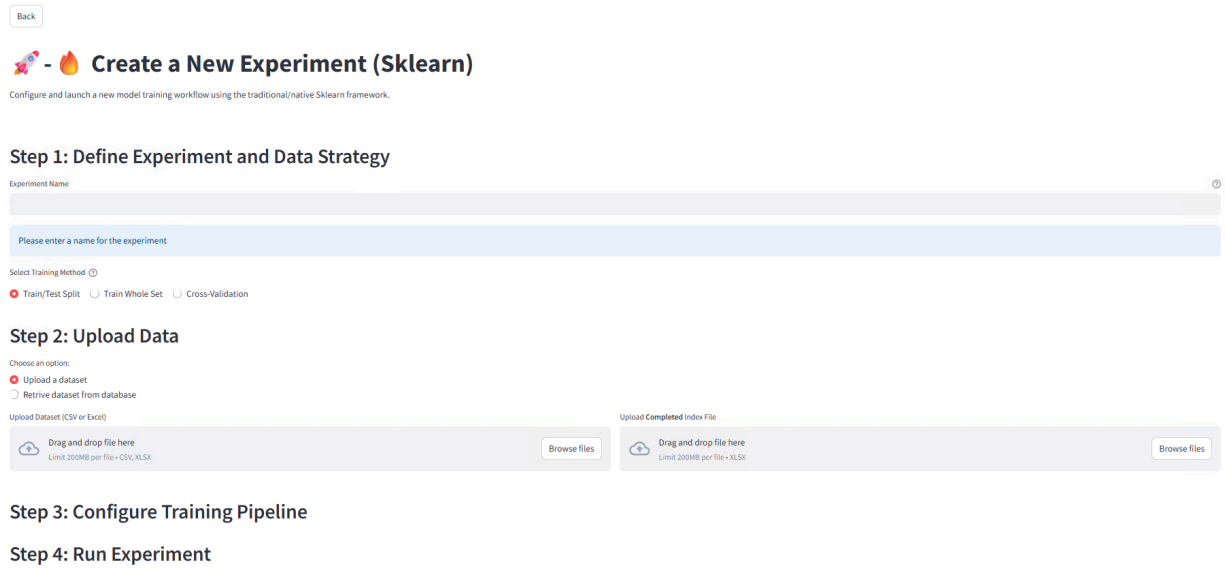
Screenshot of the option pages for the training section



When the user clicks on the training button in the main page, the screenshot above is what they will first see. The user must choose which type of model they want to train with, either with Scikit-learn/Native models or Autogulon models. There is also a back button on the top left corner where the user can go back to the main page

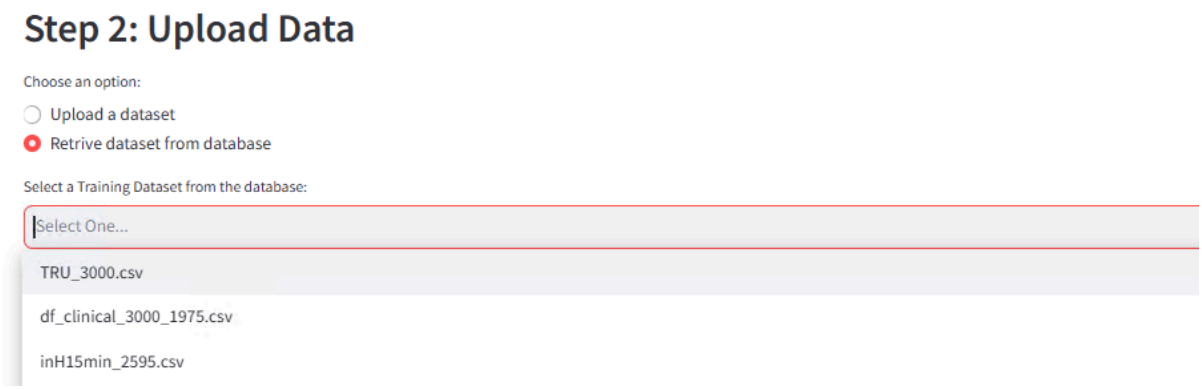
.

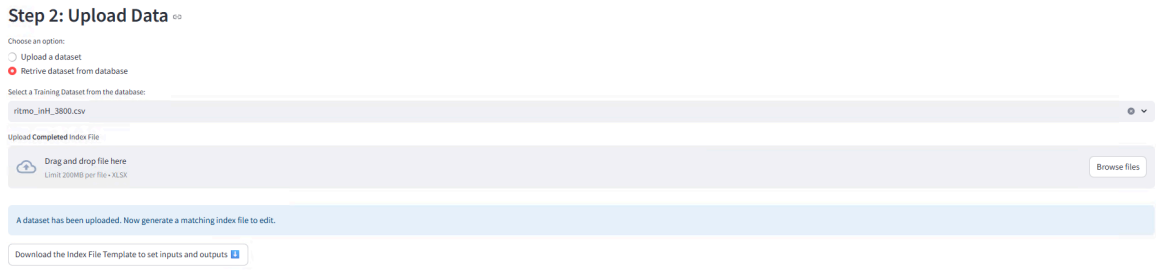
Screenshot of sklearn/scikit-learn training page

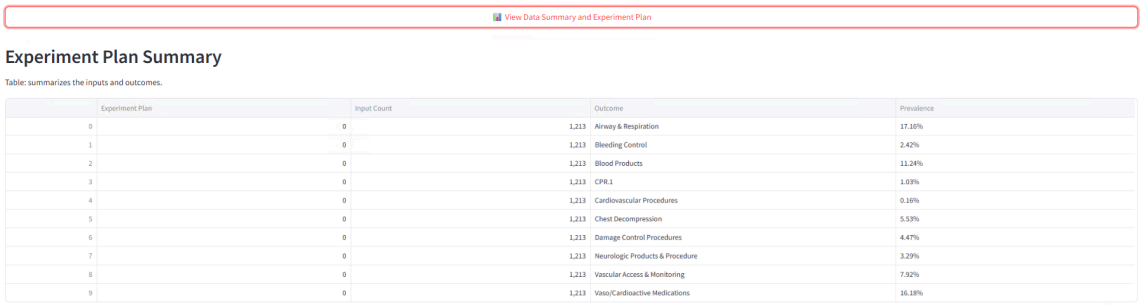


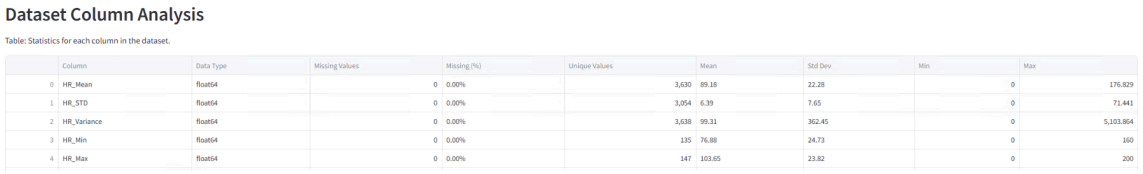
When the user clicks on the Sklearn training page, they must first name the new ML experiment they want to conduct. They must enter that name in the textbox under the “Step 1: Define Experiment and Data Strategy” header. The name must be unique, or else any further actions will be blocked if it already exists in the database. After that, the user must choose one of three options under “Select Training Method” on what kind of ML experiment they want to do, as presented on the page after the naming textbox. Train/test split, train the whole dataset, or cross-validation.

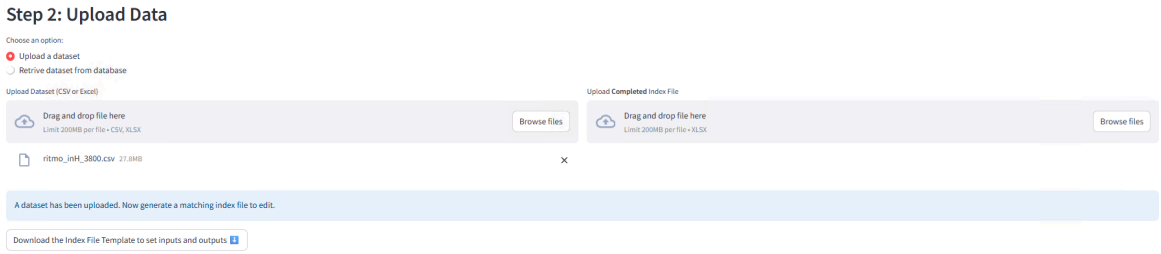
Screenshots of the dataset uploading section

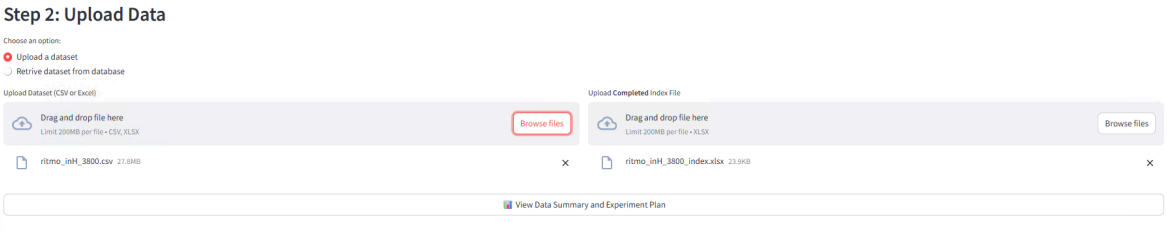




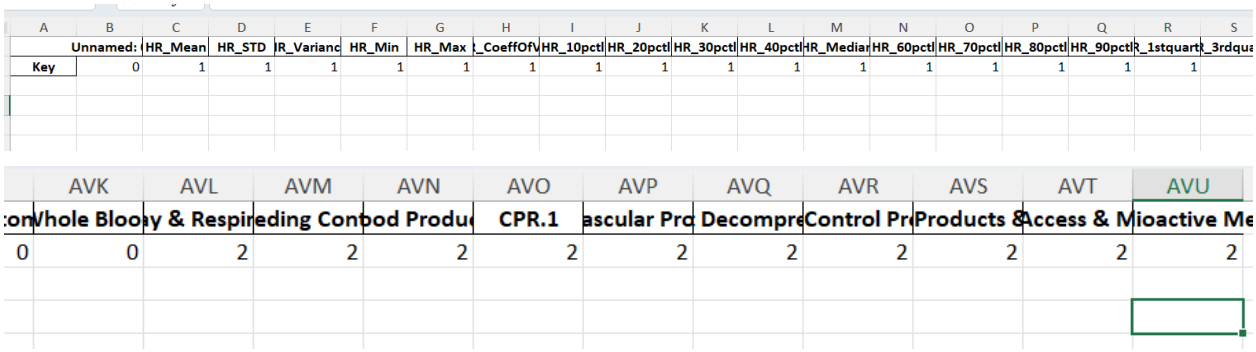








Screenshots of an index file (1=input variable, 2=output variable, 0=ignore)



	Unnamed:	HR_Mean	HR_STD	R_Varianc	HR_Min	HR_Max	_CoeffOfV	HR_10pctl	HR_20pctl	HR_30pctl	HR_40pctl	HR_Median	HR_60pctl	HR_70pctl	HR_80pctl	HR_90pctl	R_1stquartl	_3rdqua
Key	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

	AVK	AVL	AVM	AVN	AVO	AVP	AVQ	AVR	AVS	AVT	AVU
:on	Whole Bloo	y & Respi	reding Con	od Produ	CPR.1	ascular Pro	Decompre	Control Pr	Products &	Access & M	ioactive Me
0	0	2	2	2	2	2	2	2	2	2	2

Following the naming step, the user must upload a dataset to be used for training. They can either upload a dataset from the browser or retrieve a saved dataset from the database. When the user uploads a dataset and executes training, that uploaded dataset will be saved in the database for future use.

After the user uploads a dataset, they will be shown a button that will download an index file that will have the same columns as the dataset, but with only one row. That one row will have values that will indicate which variables of the dataset are the input or output variables. 1=input variable, 2=output variable, 0=ignore variable. The user must enter one of these three values for each column to indicate their use. After the user finishes customizing the index file, they then must upload it back into the app in the “upload index file” section.

When both the dataset and index file are uploaded, the user can then click on the “View Data Summary and Experiment Plan” tab, which will show the statistics of each output and input variable, including mean and standard deviation for input variables and frequency of positive values for output variables.

Screenshots of the training page if the user opts for uploading a configuration file

Step 3: Configure Training Pipeline

Choose an option:

- Upload a file
- User Customization

Enter the minimum unique value threshold for a numerical input variable to be consired catagorical:

10

Enter the number of models:

5

Download Configuration File Template

Guide for setting up configuration file

List of ML Algorithims to use.

Setup Options.

Upload a Configuration File

Drag and drop file here
Limit 10GB per file

Browse files

Step 4: Run Experiment

Guide for setting up configuration file

List of ML Algorithims to use.

Please use the short versions of the algo names shown on the **right**.


```
{
  "Random Forest" : "rf"
  "XGBoost" : "xgb"
  "Cat Boost" : "cat"
  "SGD Elastic" : "sgd_elastic"
  "SGD L2" : "sgd_l2"
  "Logistic Reg. L2" : "lr_l2"
  "Logistic Reg." : "lr"
  "Descision Tree" : "dt"
  "SVM" : "svm"
  "KNearest N." : "knn"
}
```

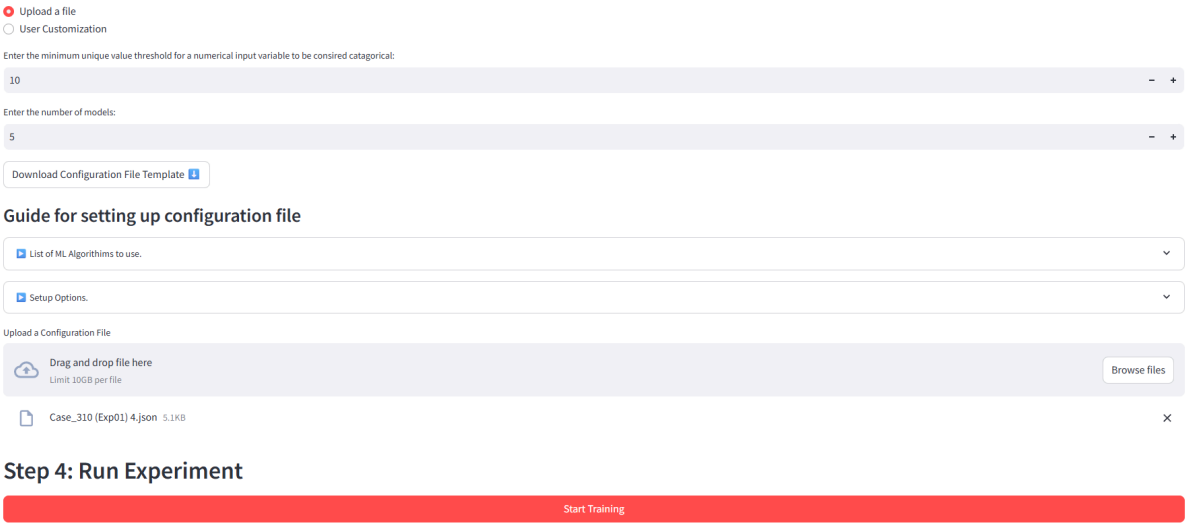

Guide for setting up configuration file

List of ML Algorithims to use.

Setup Options.

Please use the names shown on the **lists**.


```
{
  "impute_strategy" : "['mean', 'median', 'most_frequent', 'constant']"
  "cut threshold (range)" : "(0.0 - 1.0)"
  "inf (how to handle inf values in data)" : "['replace with null', 'replace with zero']"
  "outliers (how to handle outliers in data)" : "['None', 'remove rows', 'log']"
  "scalingMethod" : "['MinMaxScaler', 'RobustScaler', 'MaxAbsScaler', 'StandardScaler']"
  "rebalance_type" : "['RandomUnderSampler', 'RandomOverSampler', 'SMOTE', 'ADASYN']"
  "sampling_strategy" : "['auto', 'majority', 'not minority', 'not majority', 'all', 'ratio']"
  "FeatureSelection_method" : "['MRMR', 'RFECV', 'SelectKBest-f_classif', 'SelectKBest-chi2']"
  "strategy" : "['random', 'bayesian', 'grid']"
}
```



After uploading the training set and its index file customized, the user then sets up the machine learning configuration, which they do in the section on the page under “Step 3: Configure Training Pipeline”. The user can choose to either download a configuration file template via the “Upload a file” option and customize it or use the user interface to set up the configuration via the “User customization” option.

If the user chooses to upload a configuration file, they will be asked to enter the minimum unique value threshold for an input variable to be considered a categorical variable and the number of models they want to train. After that, they must click on the “Download Configuration File Template” to download a configuration file that has a set for each of the n models the user requests. The “List of ML Algorithims to use.” tab shows the available ML algorithms the user can use for each model.

Screenshots of the configuration file before and after customization

```
{
    "Eric's Sklearn Test": {
        "threshold_type": "enter_type_here",
        "exp_0": {
            "algorithm": "enter_algo_here",
            "options": {
                "oneHotEncode": "True",
                "Impute": "True",
                "cutMissingRows": "True",
                "cut threshold": 0.6,
                "inf": "replace with null",
                "outliers": "log",
                "outliers_N": 50000,
                "Scaling": "True",
                "scalingMethod": "StandardScaler",
                "QuantileTransformer": "True",
                "Normalize": "True",
                "rebalance": "True",
                "rebalance_type": "SMOTE",
                "FeatureSelection": "True",
                "method": "SelectKBest-f_classif",
                "N_features": 40,
                "strategy": "random",
                "itr": 50,
                "CV": 10,
                "n_repeats": 1,
                "min_postives": 10,
                "test_size": 0.2
            },
            "param_vals": "None"
        },
        "exp_1": {
            "algorithm": "enter_algo_here",
            "options": {
                "oneHotEncode": "True",
                "Impute": "True",
```

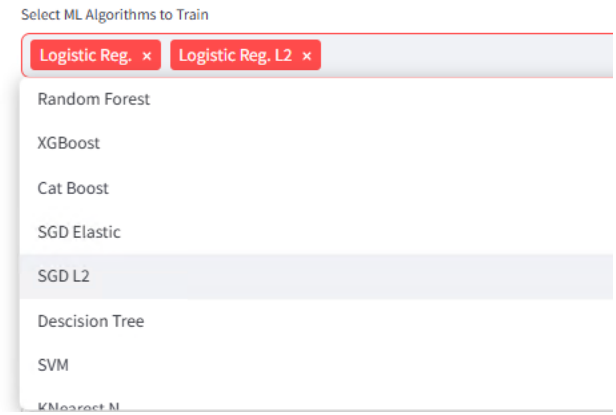
```
{
    "Eric's Sklearn Test": {
        "threshold_type": "youden",
        "exp_0": {
            "algorithm": "lr",
            "options": {
                "oneHotEncode": "True",
                "Impute": "True",
                "cutMissingRows": "False",
                "cut threshold": 0.6,
                "inf": "replace with null",
                "outliers": "log",
                "outliers_N": 60000,
                "Scaling": "True",
                "scalingMethod": "StandardScaler",
                "QuantileTransformer": "False",
                "Normalize": "False",
                "rebalance": "False",
                "rebalance_type": "SMOTE",
                "FeatureSelection": "True",
                "method": "SelectKBest-f_classif",
                "N_features": 40,
                "strategy": "random",
                "itr": 50,
                "CV": 10,
                "n_repeats": 1,
                "min_postives": 10,
                "test_size": 0.2
            },
            "param_vals": "None"
        },
        "exp_1": {
            "algorithm": "lr_l2",
            "options": {
                "oneHotEncode": "True",
                "Impute": "True",
                "cutMissingRows": "True",
```

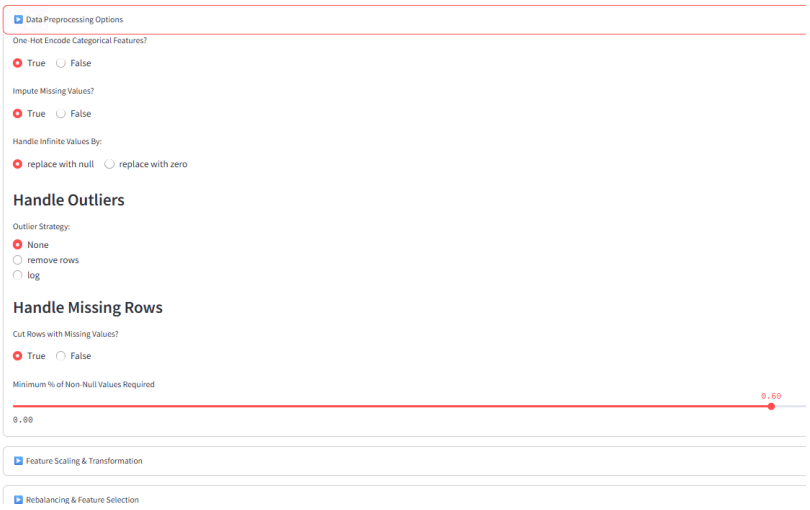
```
"exp_4": {
    "algorithm": "xgb",
    "options": {
        "oneHotEncode": "True",
        "Impute": "True",
        "cutMissingRows": "True",
        "cut threshold": 0.60,
        "inf": "replace with null",
        "outliers": "log",
        "outliers_N": 50000,
        "Scaling": "True",
        "scalingMethod": "StandardScaler",
        "QuantileTransformer": "False",
        "Normalize": "False",
        "rebalance": "False",
        "rebalance_type": "SMOTE",
        "FeatureSelection": "True",
        "method": "SelectKBest-f_classif",
        "N_features": 100,
        "strategy": "random",
        "itr": 125,
        "CV": 10,
        "n_repeats": 10,
        "min_postives": 10,
        "test_size": 0.2,
        "cutoff_index": "youden"
    },
    "param_vals": {"max_depth": [2, 3, 4, 5, 6],
    "n_estimators": [10, 15, 20, 25, 30, 40, 50, 80, 100, 150, 200],
    "learning_rate": [0.00001, 0.0001, 0.0005, 0.001, 0.005, 0.01],
    "subsample": [0.5, 0.7, 0.8, 0.9, 1.0],
    "reg_lambda": [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
    "colsample_bytree": [0.6, 0.7, 0.8, 0.9, 1.0],
    "gamma": [0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}
}
```

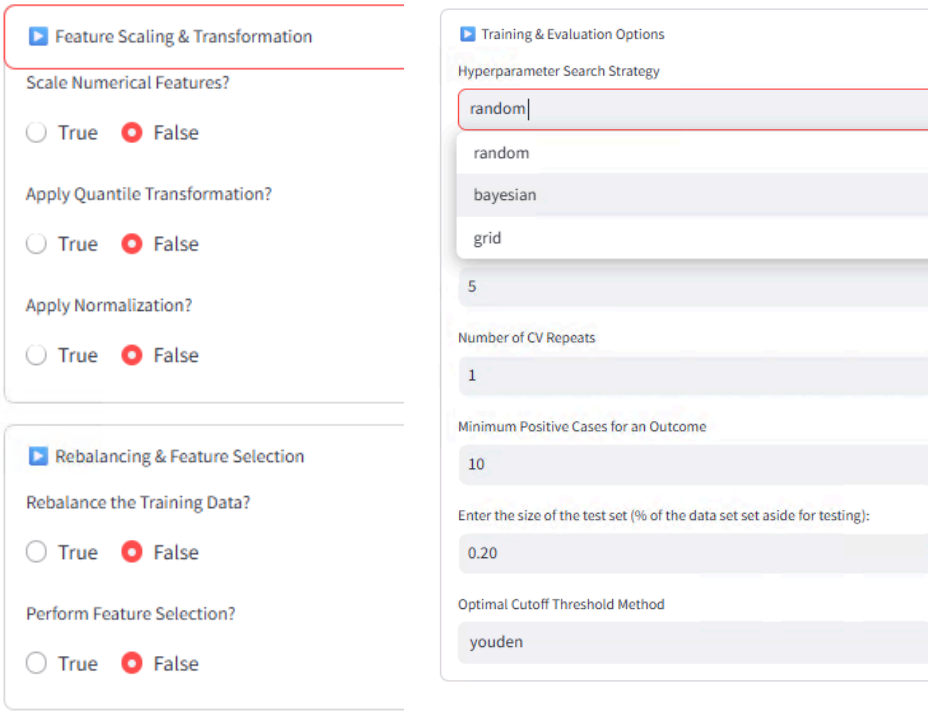
After the user customizes the configuration file, they must upload it back into the app via the “Upload a Configuration File” section.

The advantage of the “Upload a file” option is that users can make the ML setup for each individual model unique from one another and can provide a feature space for hyperparameter tuning. This provides more flexibility.

Screenshots of the training page if the user opts for customizing the training process via interface



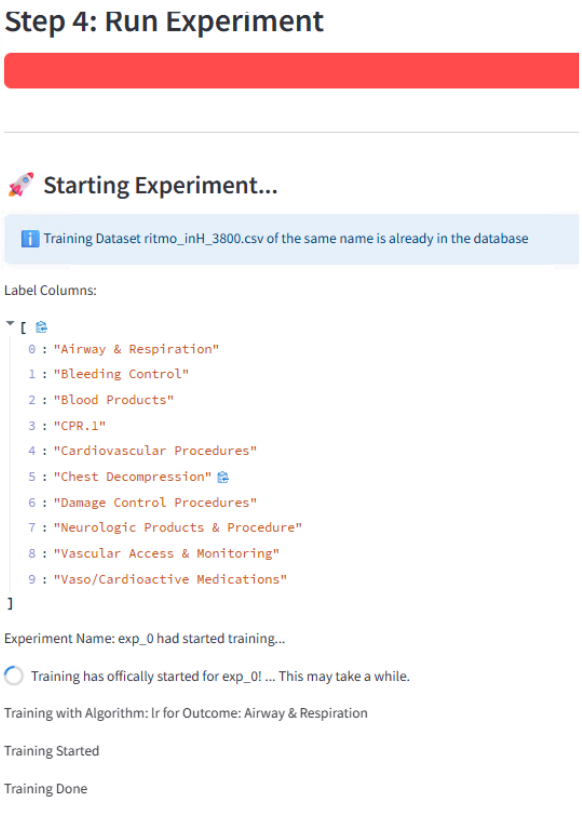




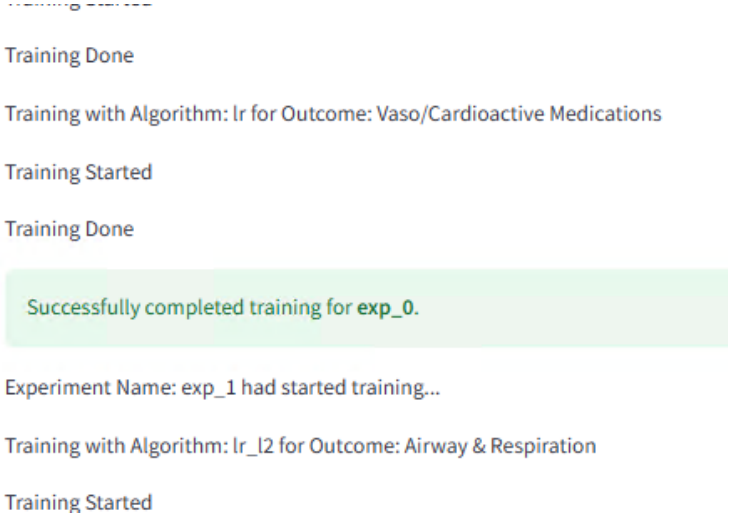
If the user chooses to customize the training process via “User customization”, they must first select any set of ML algorithms they want to train with. Next, they enter the minimum unique value threshold for an input variable to be considered a categorical variable. They will then be given 4 drop-down tabs, which each of them presents a set of options for the user to set up the machine learning training process.

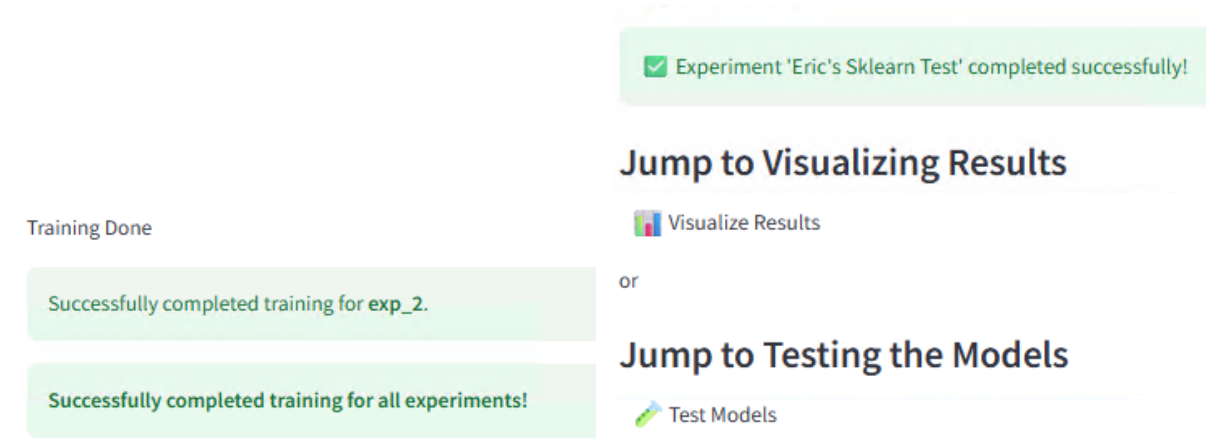
The advantage of “User customization” is that it is much easier to use and possibly take less time. This most likely results in a configuration that is simplified and uniform for all models.

Screenshots of the training in process



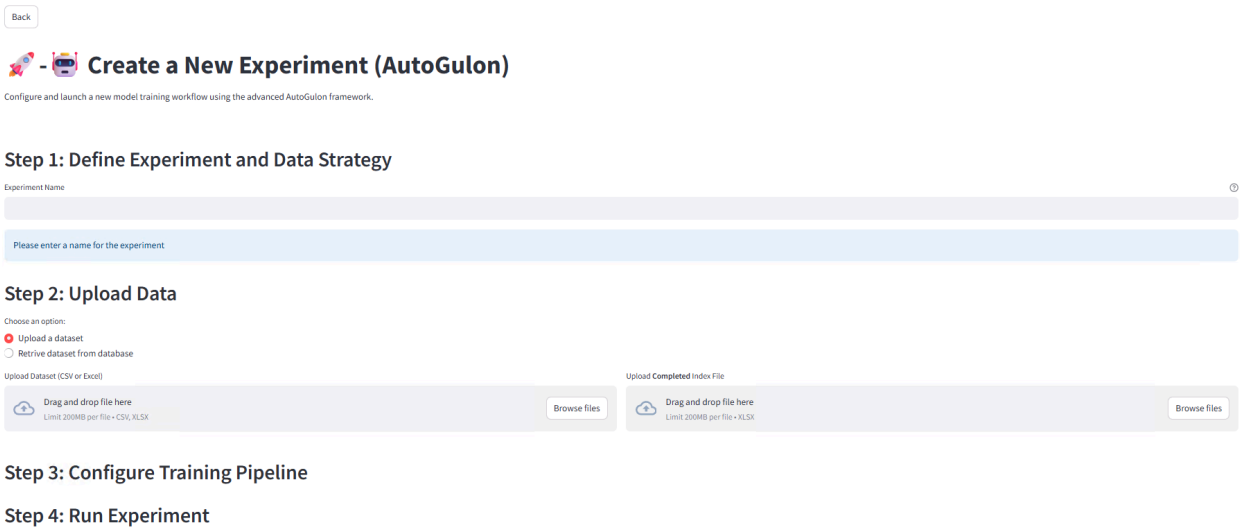
Screenshots of the training page after training is done





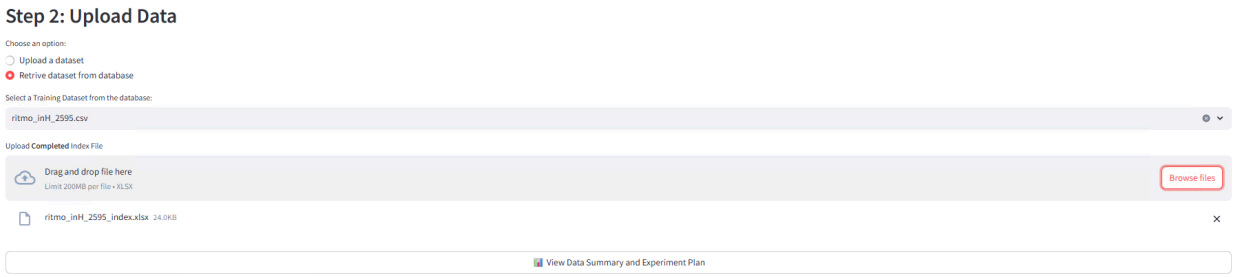
After the configuration setup is complete, the user must then click on the training “Start Training” button to start training the models. The user will then have to wait a while until the training process is completed. If it does, the user interface will show green tabs that will indicate that training is successfully completed.

Screenshot of the AutoGulon training page



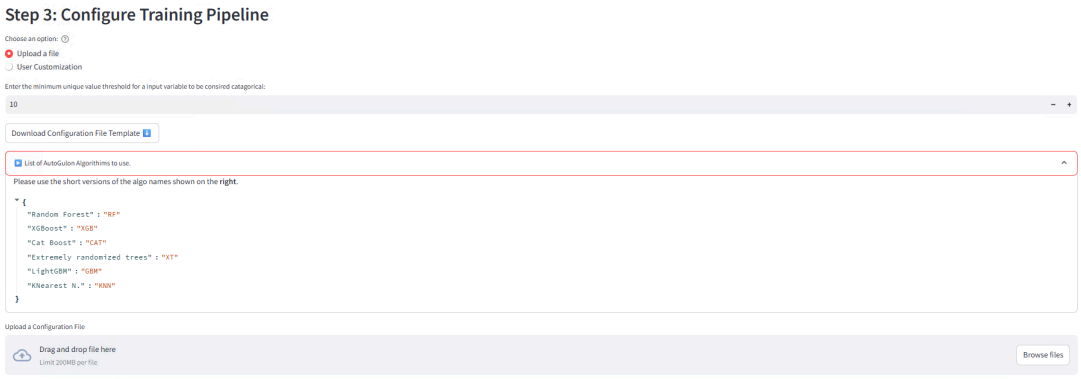
The training page for the AutoGulon section is nearly the same as the Sklearn training page. The only difference is that AutoGulon does not have multiple training type options, as in all cases, the entire training set is used for training, and you do not need to select a set of ML algorithms unless you want to customize the feature space of hyperparameter tuning.

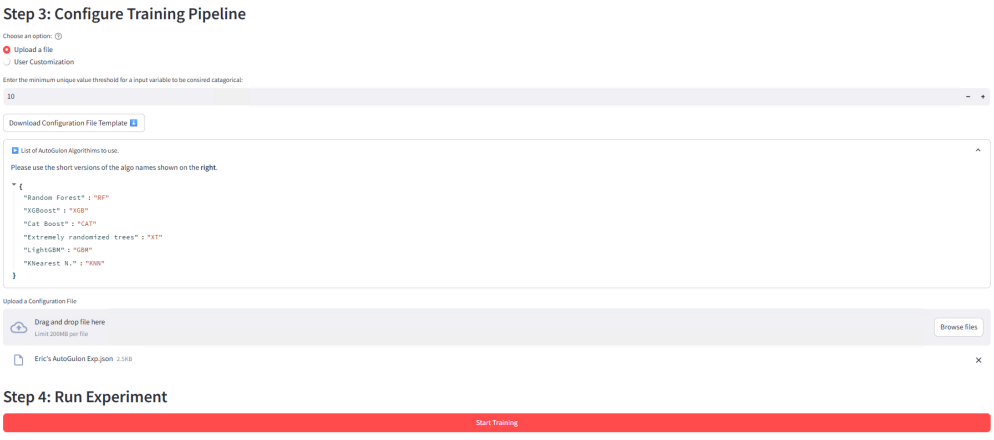
Screenshot of the dataset uploading section



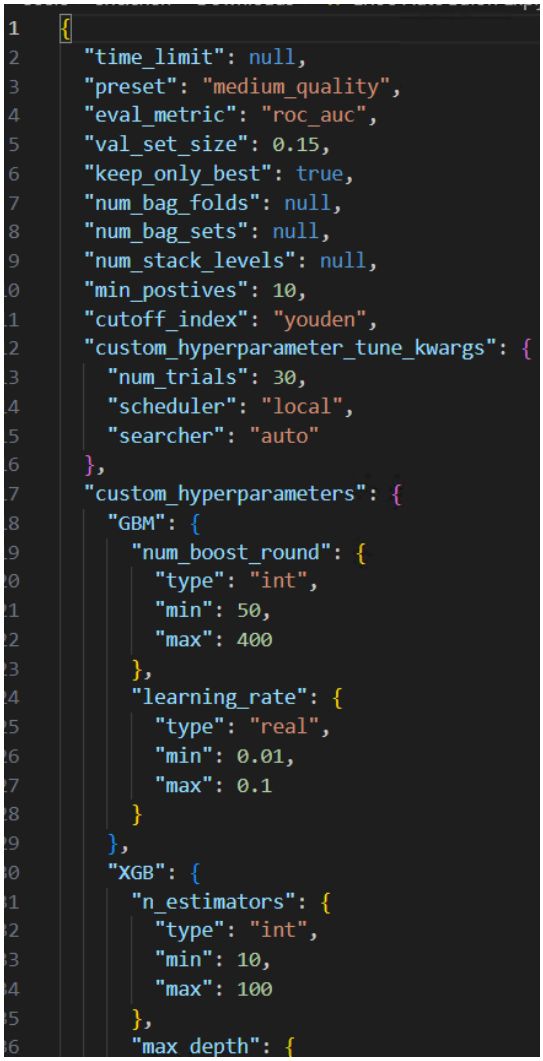
Data uploading is the same for AutoGluon training as it is for Sklearn training.

Screenshot of the training page if user opts for uploading a configuration file

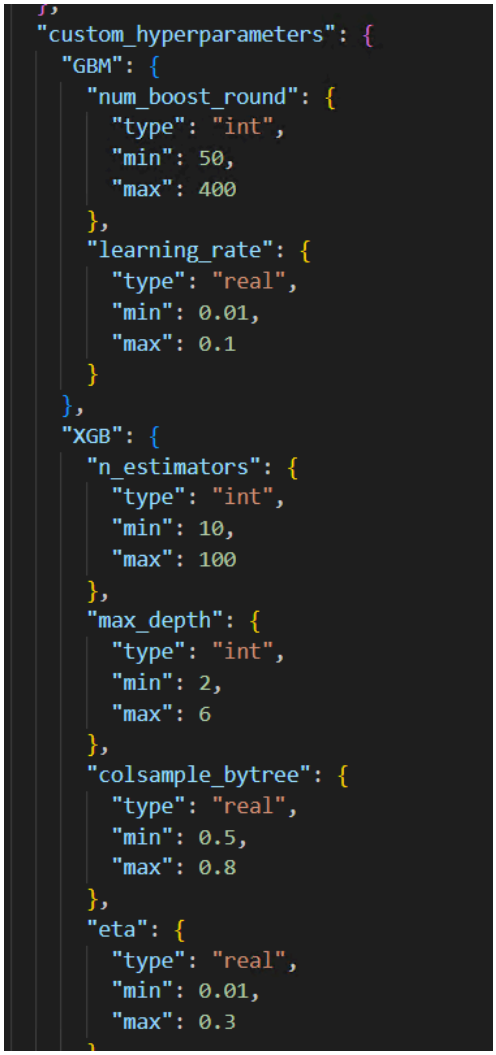




Screenshots of the configuration file



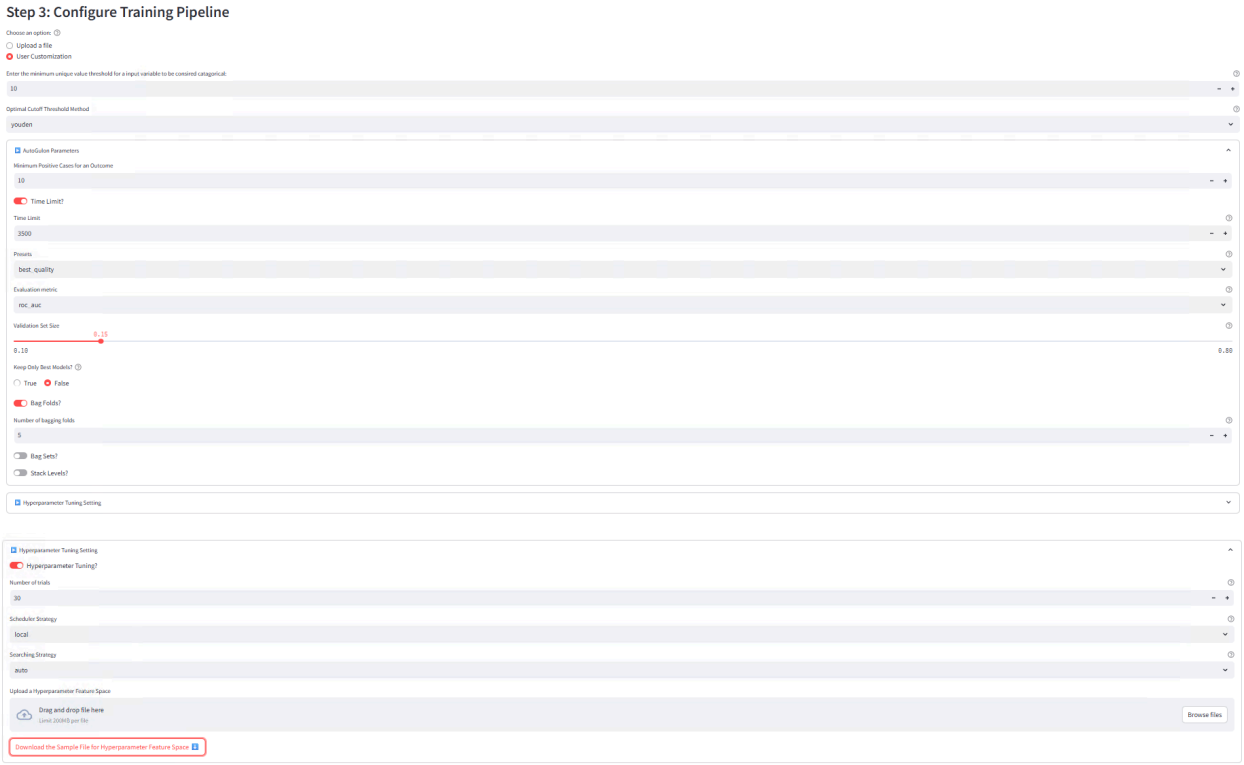
```
{
  "time_limit": null,
  "preset": "medium_quality",
  "eval_metric": "roc_auc",
  "val_set_size": 0.15,
  "keep_only_best": true,
  "num_bag_folds": null,
  "num_bag_sets": null,
  "num_stack_levels": null,
  "min_postives": 10,
  "cutoff_index": "youden",
  "custom_hyperparameter_tune_kwargs": {
    "num_trials": 30,
    "scheduler": "local",
    "searcher": "auto"
  },
  "custom_hyperparameters": {
    "GBM": {
      "num_boost_round": {
        "type": "int",
        "min": 50,
        "max": 400
      },
      "learning_rate": {
        "type": "real",
        "min": 0.01,
        "max": 0.1
      }
    },
    "XGB": {
      "n_estimators": {
        "type": "int",
        "min": 10,
        "max": 100
      },
      "max_depth": {
```



```
"custom_hyperparameters": {
  "GBM": {
    "num_boost_round": {
      "type": "int",
      "min": 50,
      "max": 400
    },
    "learning_rate": {
      "type": "real",
      "min": 0.01,
      "max": 0.1
    }
  },
  "XGB": {
    "n_estimators": {
      "type": "int",
      "min": 10,
      "max": 100
    },
    "max_depth": {
      "type": "int",
      "min": 2,
      "max": 6
    },
    "colsample_bytree": {
      "type": "real",
      "min": 0.5,
      "max": 0.8
    },
    "eta": {
      "type": "real",
      "min": 0.01,
      "max": 0.3
```

The functionality and mechanism of uploading a configuration file remain the same for AutoGulon, but the setup itself (including what parameters the user should specificity, etc.) is different.

Screenshot of the training page if the user opts for customizing the training process via interface



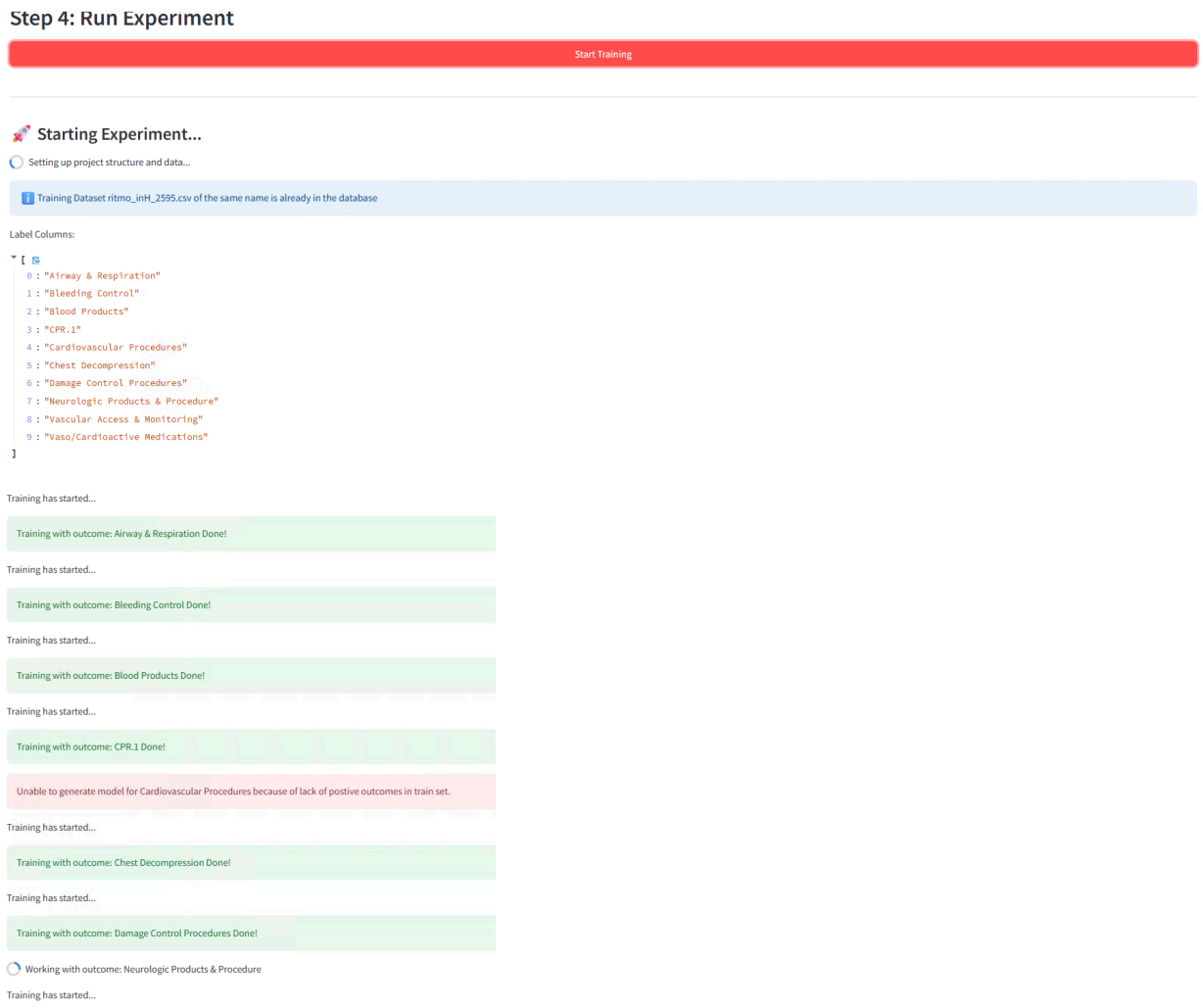
The functionality and mechanism of user customization remain the same for AutoGulon, but the setup itself and what the user will see for the customization are different.

The key difference is that if the user wants to do hyperparameter tuning, they still need to download and customize a JSON file that will provide the feature space for hyperparameter tuning.

Screenshot of the feature space file if user wants to do hyperparameter

```
{
  "GBM": {
    "num_boost_round": {
      "type": "int",
      "min": 50,
      "max": 400
    },
    "learning_rate": {
      "type": "real",
      "min": 0.01,
      "max": 0.1
    }
  },
  "XGB": {
    "n_estimators": {
      "type": "int",
      "min": 10,
      "max": 100
    },
    "max_depth": {
      "type": "int",
      "min": 2,
      "max": 6
    },
    "colsample_bytree": {
      "type": "real",
      "min": 0.5,
      "max": 0.8
    },
    "eta": {
      "type": "real",
      "min": 0.01,
      "max": 0.3
    }
  },
  "CAT": {
```

Screenshot of the training page during training



Screenshot of the training page after training is done

Just like for the Sklearn page, after the configuration setup is complete, the user must then click on the training “Start Training” button to start training the models. The user will then have to wait a while until the training process is completed. If it does, the user interface will show green tabs that will indicate that training is successfully completed.

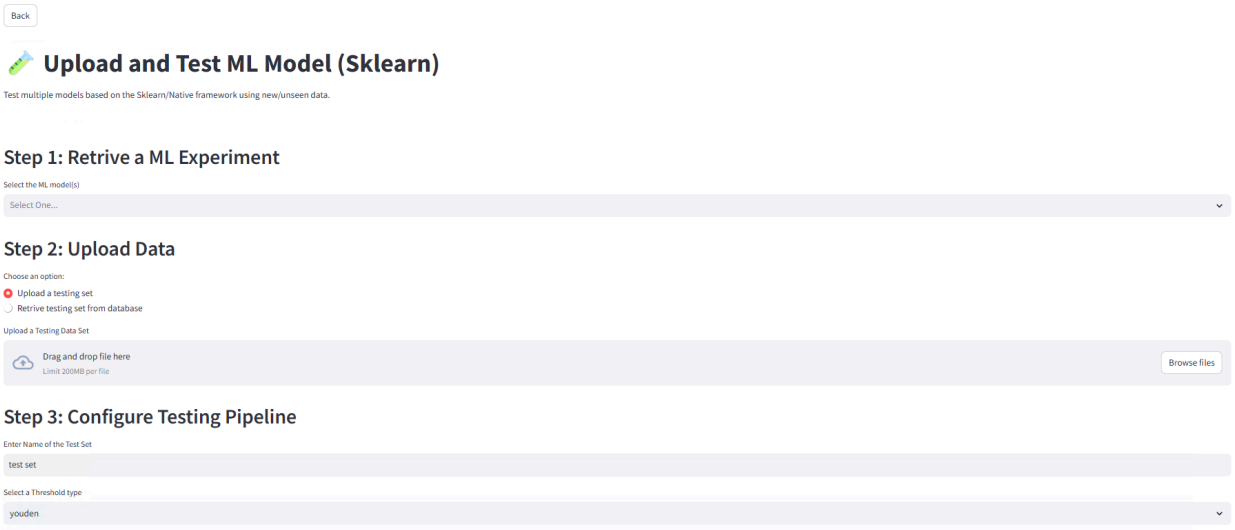
5.3 Testing

Screenshot of the option pages for the testing section

When the user first clicks on the testing button on the main page, the screenshot above is the first thing they will see. They must choose which mode of testing they are doing, depending on the type of machine learning model they intend to test on. There is also a back button on the top left corner, where the user can go back to the main page.

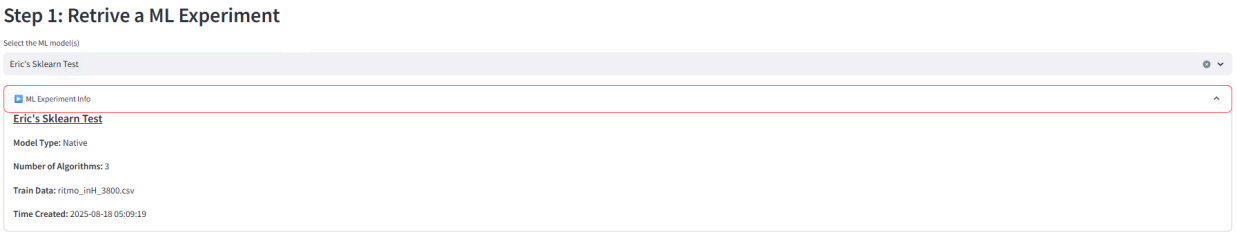
The testing page itself is virtually the same for both Sklearn and AutoGluon.

Screenshots of the testing section when first opened



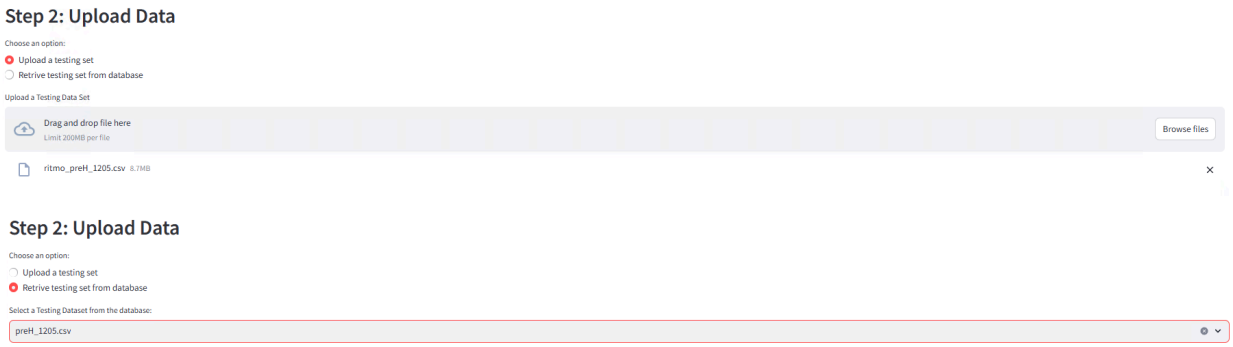
When the user first enters a testing page, the first step they must take is to select a completed ML experiment that is saved in the database via the drop-down menu under the “Step 1: Retrive a ML Experiment” header.

Screenshot of the “ML Experiment Info” dropdown tab



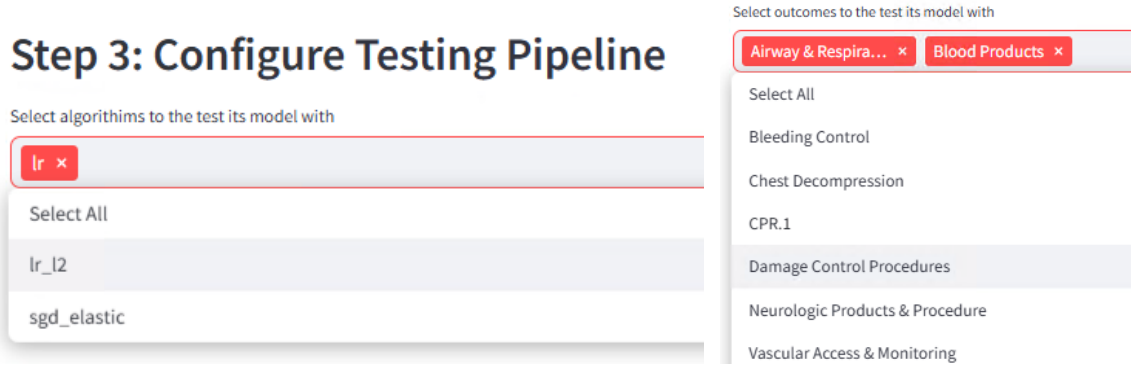
Once a user selects an ML experiment to test, they will be shown a dropdown tab called “ML Experiment Info” which will give basic information about the saved experiment. This includes the time it was created and how many algorithms it uses.

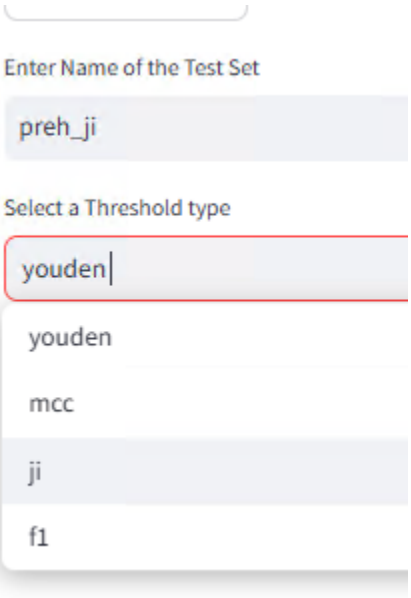
Screenshots of the dataset uploading section

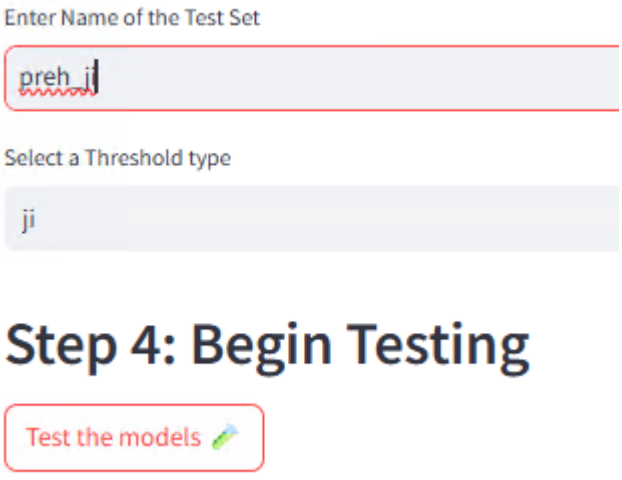


Under the “Step 2: Upload Data” header, the user must either upload a data set used for testing or retrieve one from the database. The dataset that is uploaded will be checked to see if they have the input and output variables that the ML experiment used. If they don’t, then the user will be blocked from going any further until they get a compatible dataset.

Screenshots of the configuration part

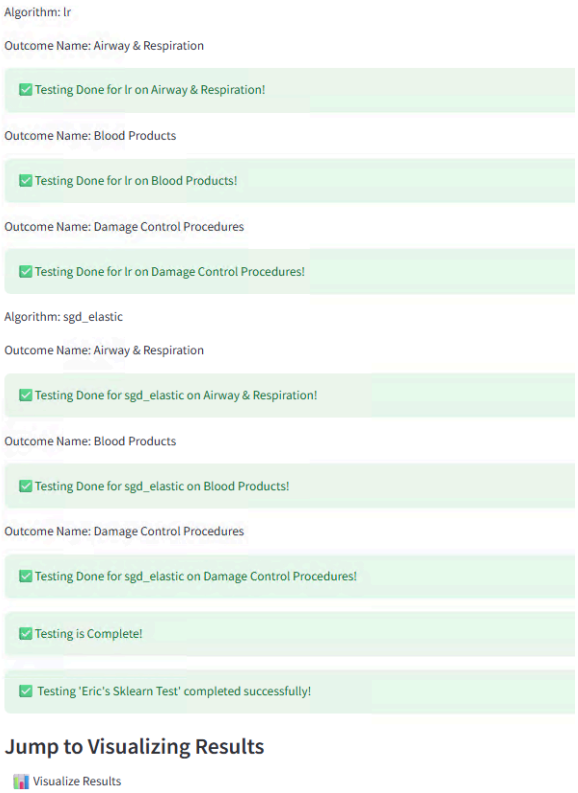






Under the “Step 3: Configure Testing Pipeline” header, the user must give a name for this test case. If the name already exists on the database, further actions will be blocked. The user also must select which set of available output variables/outcomes to be tested on (for Sklearn only) and which threshold to use for binary classification.

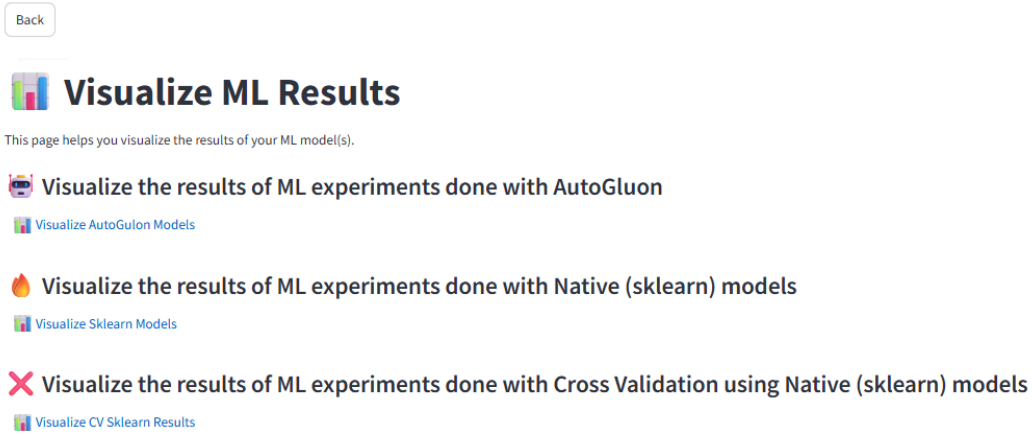
Screenshot of the testing page after testing is done

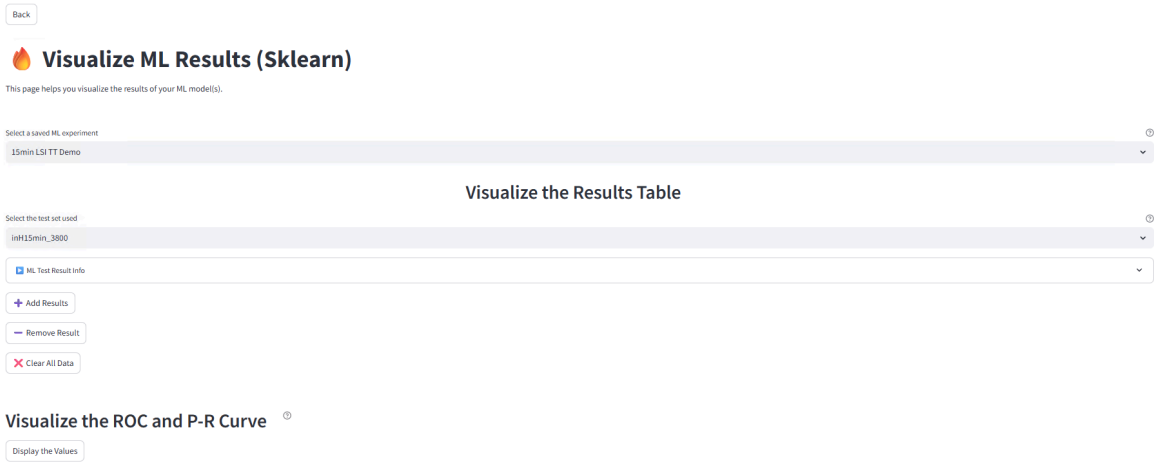


After the testing setup is complete, the user must click the “Test the models” button to start testing. The user may have to wait a while until testing is complete. If the user uploaded a test set, that test set will be saved in a database for later use. The user inference will show a green table to indicate if testing is successfully completed.

5.4 Visualization

Screenshot of the option pages for the visualization section and a visualization page once user enters one

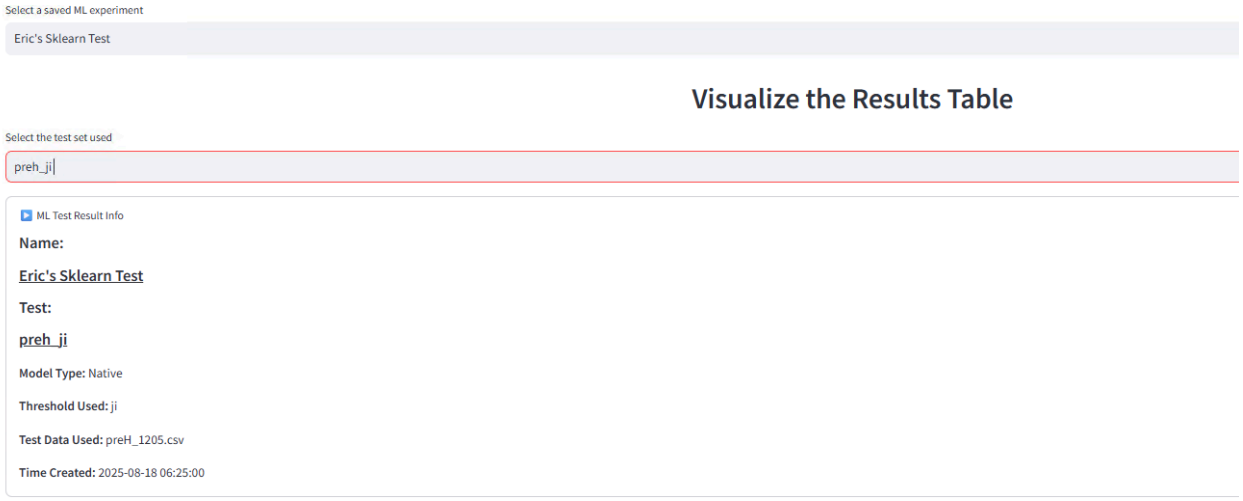




When the user first clicks on the visualization button on the main page, the screenshot above is the first thing they will see. They must choose which mode of visualization they want to use, depending on the type of machine learning model/experiment they are working on. There is also a back button on the top left corner, where the user can go back to the main page.

The visualization page itself is virtually the same for both Sklearn (both standard and cross-validation) and AutoGluon.

Screenshots of the “ML Test Result Info” section



When the user selects a saved ML experiment and a specific result from that experiment from the dropdown menus, a dropdown tab named “ML Test Result Info” will display the basic information of that test result, including the model type, the name of the test set used, the threshold used, and the time created.

Screenshots of the collective table for the visualization section

Test Data Used: ritmo_inH_3800.csv

Time Created: 2025-08-18 05:09:20

+ Add Results

− Remove Result

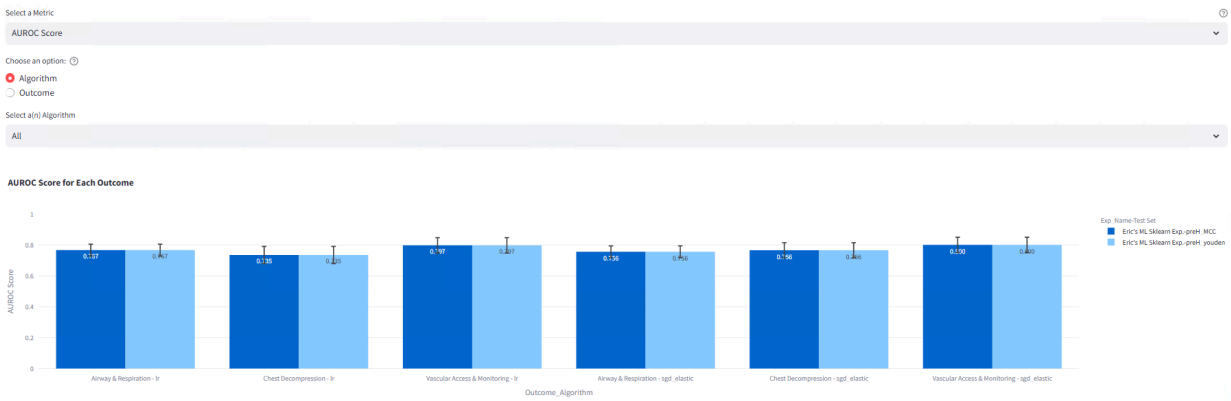
✕ Clear All Data

ML Results Table:

	Exp_Name	Test Set	Outcome	Algorithm	AUROC Score	AUROC CI Lower	AUROC CI Upper	Accuracy	Precision	Recall	TPR	TNR
0	Eric's Sklearn Test	preh_ji	Airway & Respiration	lr	0.7667	0.7281	0.8053	0.8398	0.6385	0.5397	0.5397	0.9192
1	Eric's Sklearn Test	preh_ji	Blood Products	lr	0.7905	0.7525	0.8284	0.8556	0.4907	0.462	0.462	0.9207
2	Eric's Sklearn Test	preh_ji	Damage Control Procedures	lr	0.7033	0.6353	0.7712	0.9029	0.2045	0.2769	0.2769	0.9386
3	Eric's Sklearn Test	preh_ji	Airway & Respiration	sgd_elastic	0.75	0.7119	0.788	0.8141	0.5556	0.5556	0.5556	0.8825
4	Eric's Sklearn Test	preh_ji	Blood Products	sgd_elastic	0.7972	0.7606	0.8338	0.81	0.3876	0.5848	0.5848	0.8472
5	Eric's Sklearn Test	preh_ji	Damage Control Procedures	sgd_elastic	0.705	0.6387	0.7714	0.8954	0.1856	0.2769	0.2769	0.9307
0	Eric's Sklearn Test	ritmo_inl	Airway & Respiration	lr	0.7621	0.7144	0.8097	0.8079	0.4494	0.5462	0.5462	0.8619
1	Eric's Sklearn Test	ritmo_inl	Bleeding Control	lr	0.7462	0.6347	0.8576	0.7592	0.0588	0.6111	0.6111	0.7628
2	Eric's Sklearn Test	ritmo_inl	Blood Products	lr	0.7632	0.7065	0.8199	0.7342	0.2489	0.6824	0.6824	0.7407
3	Eric's Sklearn Test	ritmo_inl	CPR.1	lr	0.7538	0.5854	0.9223	0.4526	0.0189	1	1	0.4468

When the user selects a saved ML experiment and a specific result from that experiment from the dropdown menus, they can click on the “Add Results” button to add its table result to the collective results table. They can also click on the “Remove Result” button to remove that same experiment’s results from the collective results table if it is there. The “Clear All Data” button will clear the entire table. The user can add as many unique ML results to the collective table as they please. All ML results that are added to the collective table will be represented in the bar chart and other charts shown below the table.

Screenshots of the bar chart for the visualization section

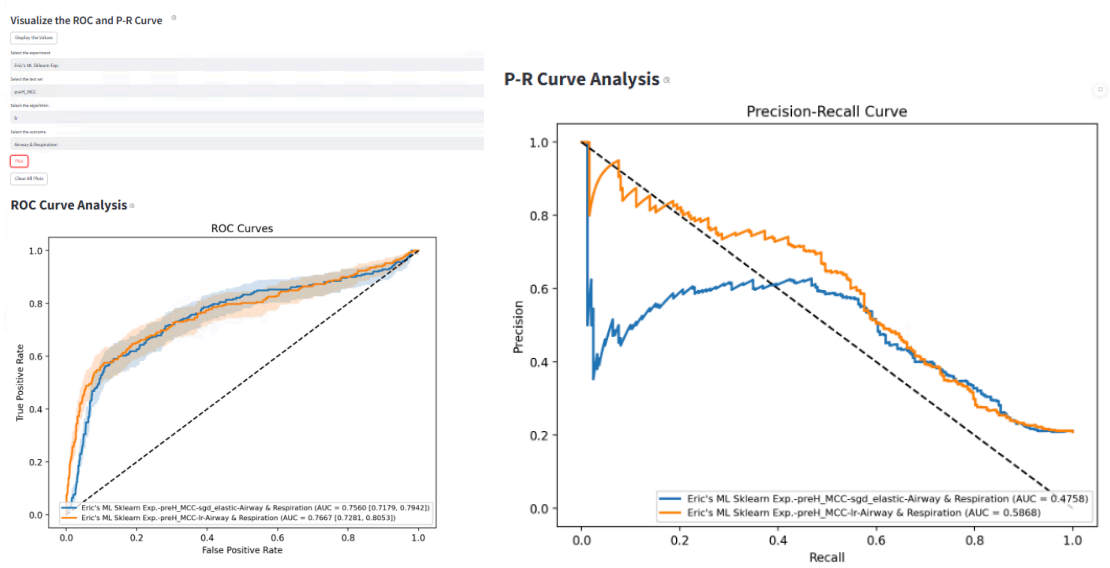


The group bar chart takes in all ML results added to the table above and then presents all the scores for a user-selected metric score (such as AU-ROC) for every one of those ML results. The bars are color-coded by which ML experiment and test result they came from and can be grouped by either algorithm or outcome, depending on the user's choice. In other words, the

user can compare metric scores from each experiment, filtered by algorithms (for Sklearn) or outcomes chosen by the user.

The bar chart at first will show every single available metric score for each outcome, on every test case, and every ML experiment. This may lead to an ugly or messy bar chart at the beginning. However, if the user selects either outcome or algorithm (for Sklearn) and chooses which outcome or algorithm to filter by, then the barchart will become more neat and will show only metrics for the specified outcome or algorithm (for Sklearn).

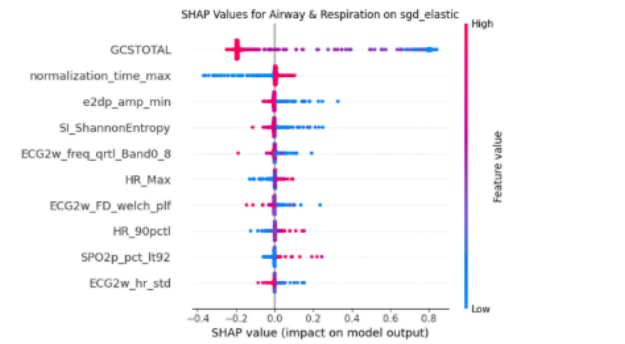
Screenshots of the ROC and P-R plot chart for the visualization section



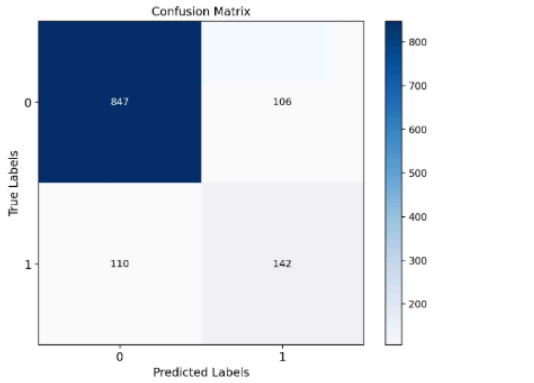
Under the header, “Visualize the ROC and P-R Curve”, the user can choose a specific ML experiment, test case, algorithm (for Sklearn), and outcome to plot an ROC and P-R curve for that outcome for that experiment’s test. The user can do this multiple times to plot as many ROC and P-R curves as they please. With that, they can compare their area-under-curve scores among multiple results and/or multiple experiments. (Ex: Compare the AU-ROC for outcome “Bleeding Control” among all models trained with Xgboost from three different experiments)

Screenshots of the SHAP/Confusion Matrix/Feature Importance chart for the visualization section

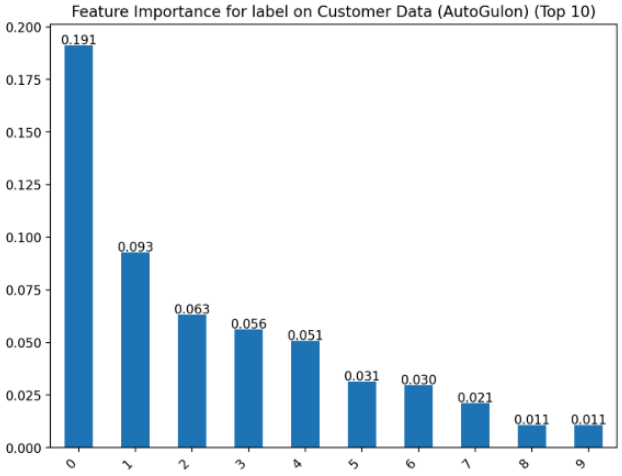
SHAP Value Analysis

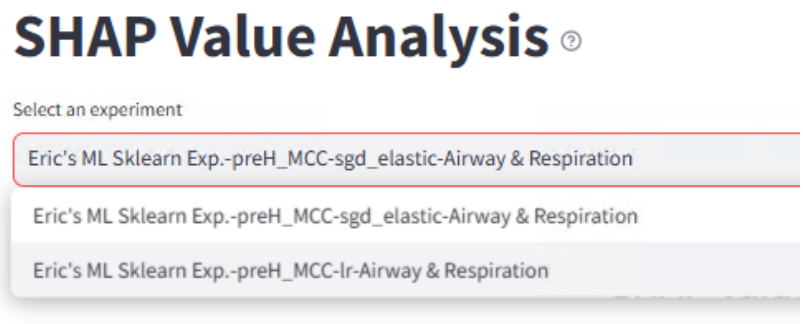


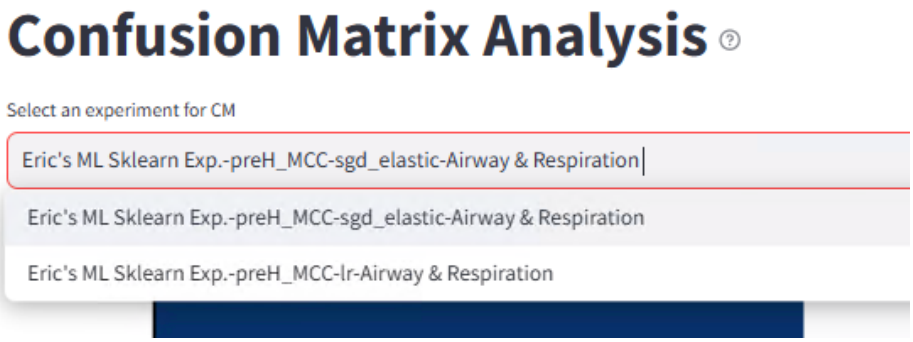
Confusion Matrix Analysis



Feature Importance Analysis



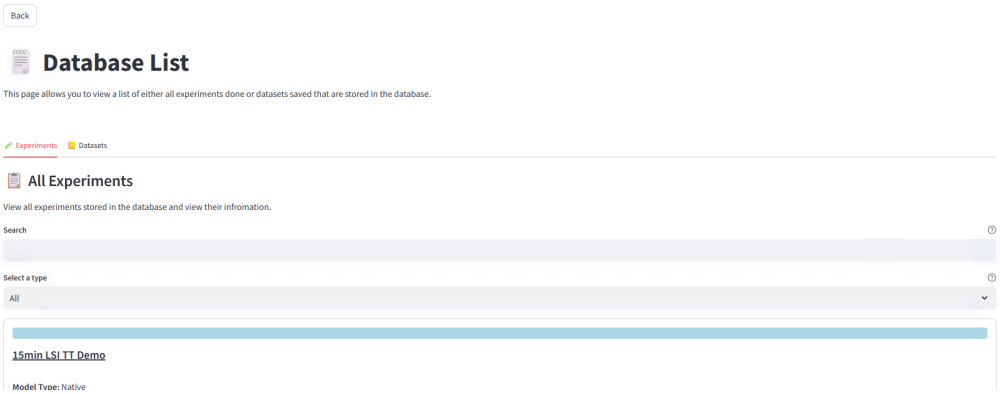




For each ROC/P-R curve plotted, the user also sees its associated SHAP chart or Feature Importance, and Confusion Matrix Analysis. Use the dropdown menu to toggle between different outcomes that were plotted in the ROC/P-R chart.

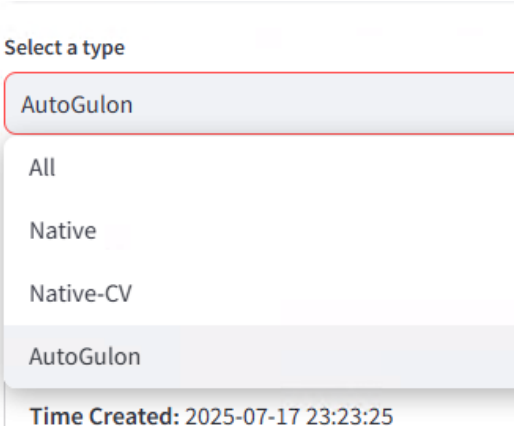
5.5 Database List

Screenshot of the listing section



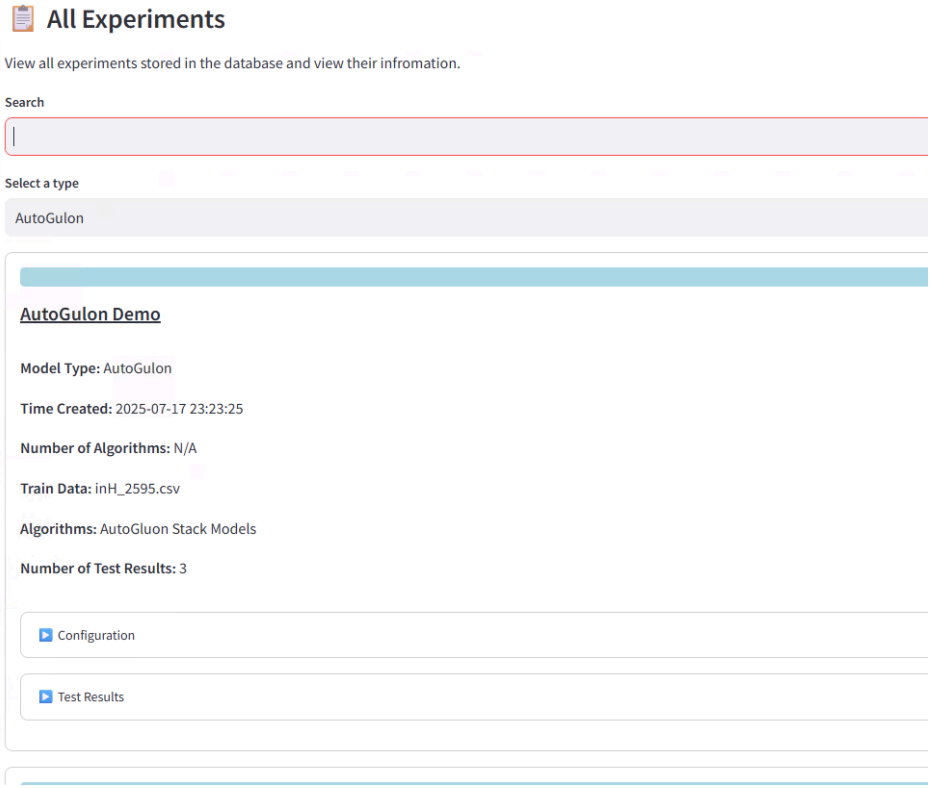
When the user first clicks on the database list button in the main page, the screenshot above is the first thing they will see. They can switch between two tabs, either for experiments or saved datasets, in which they can see a list of all of them with their metadata. The mechanism for both is the same. There is also a back button on the top left corner, where the user can go back to the main page.

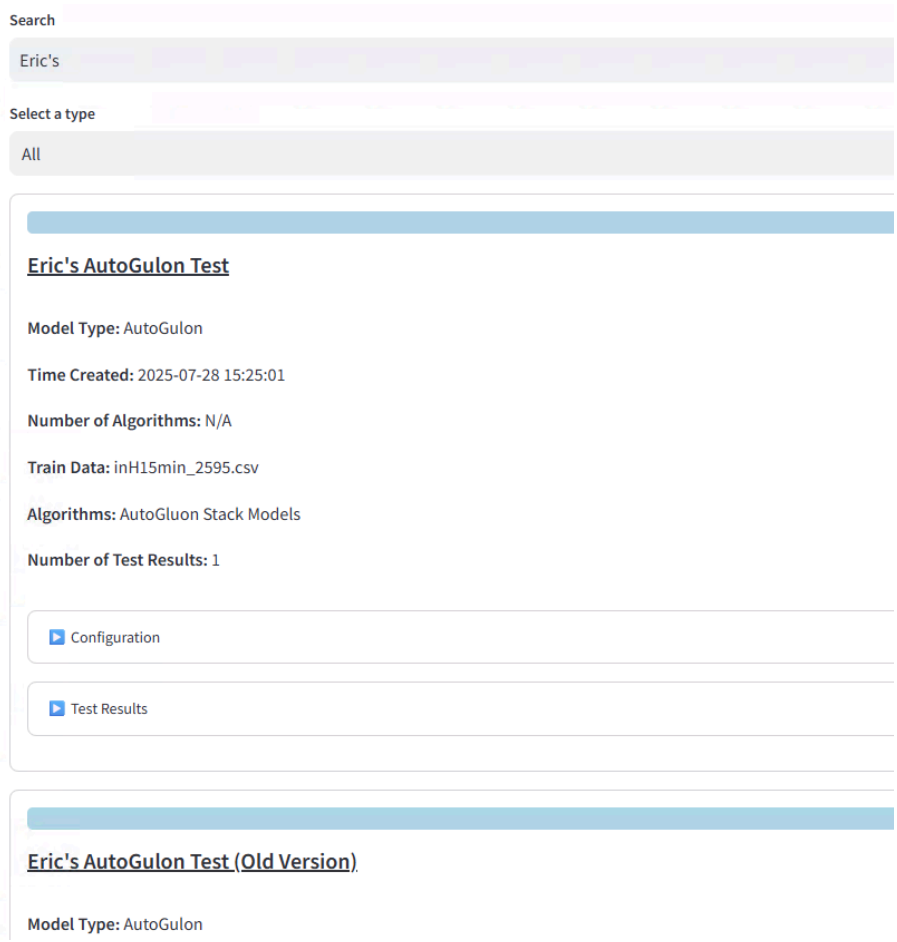
Screenshots of the “Select the type” option



The user can use the dropdown menu under “Select a type” to filter the type of ML experiments or the type of dataset they want to see. The user can also use the search bar to find a specific experiment or dataset they want to see by name.

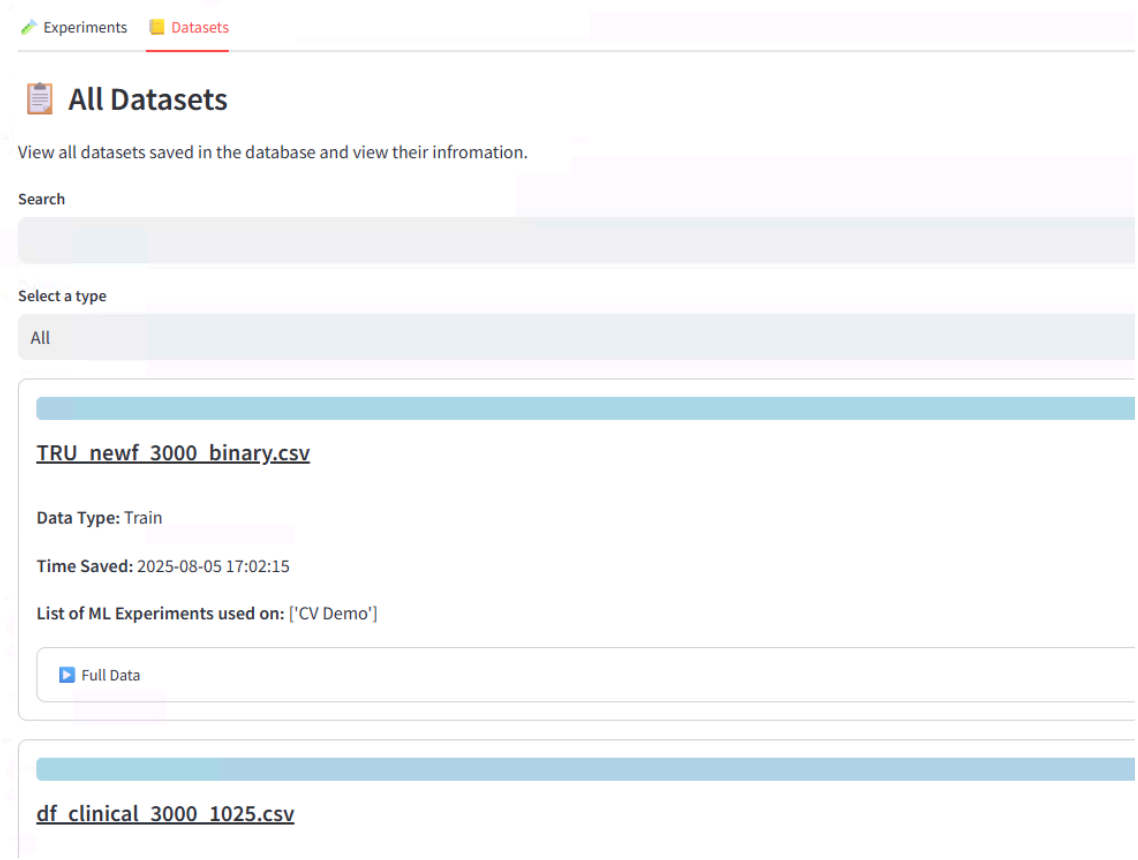
Screenshots of the experiment list





For each ML experiment listed, the user can see its name, the time created, the number and list of algorithms used, the name of the training set used, the configuration, and the number of and the results tables of test sets done for this experiment.

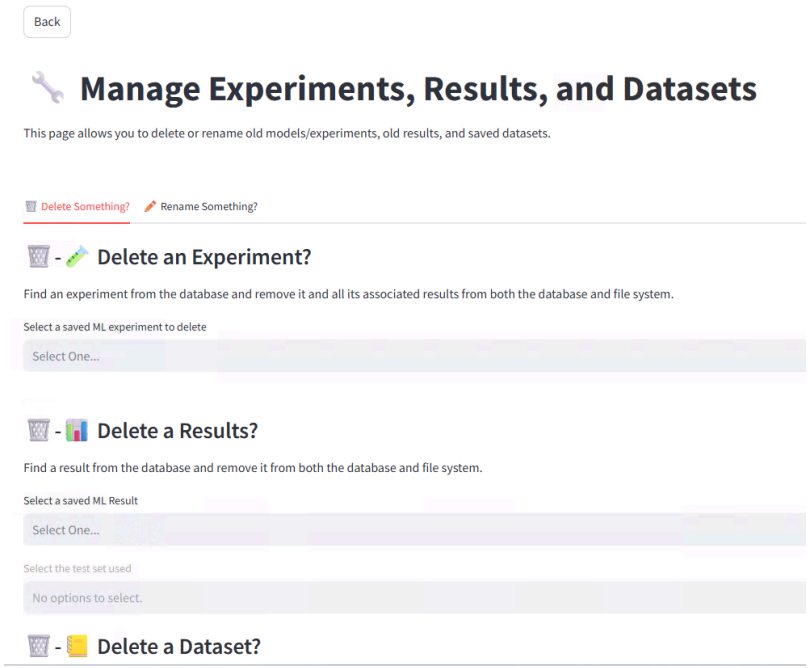
Screenshots of the database list



For each dataset saved listed, the user can see its name, the time saved, the list of all ML experiments it was used on, and the data itself.

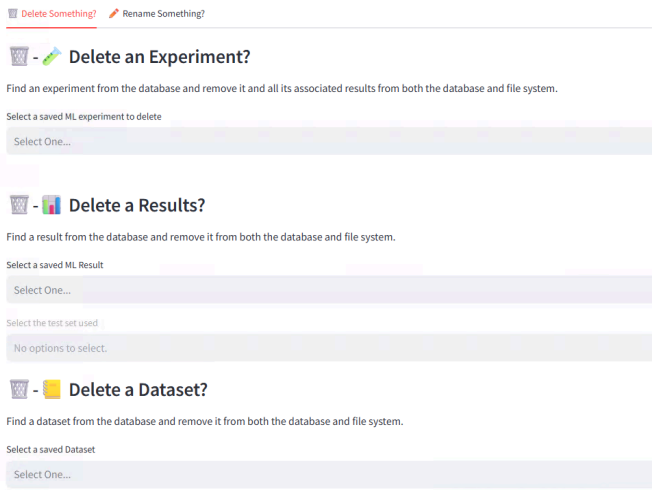
5.6 Deletion

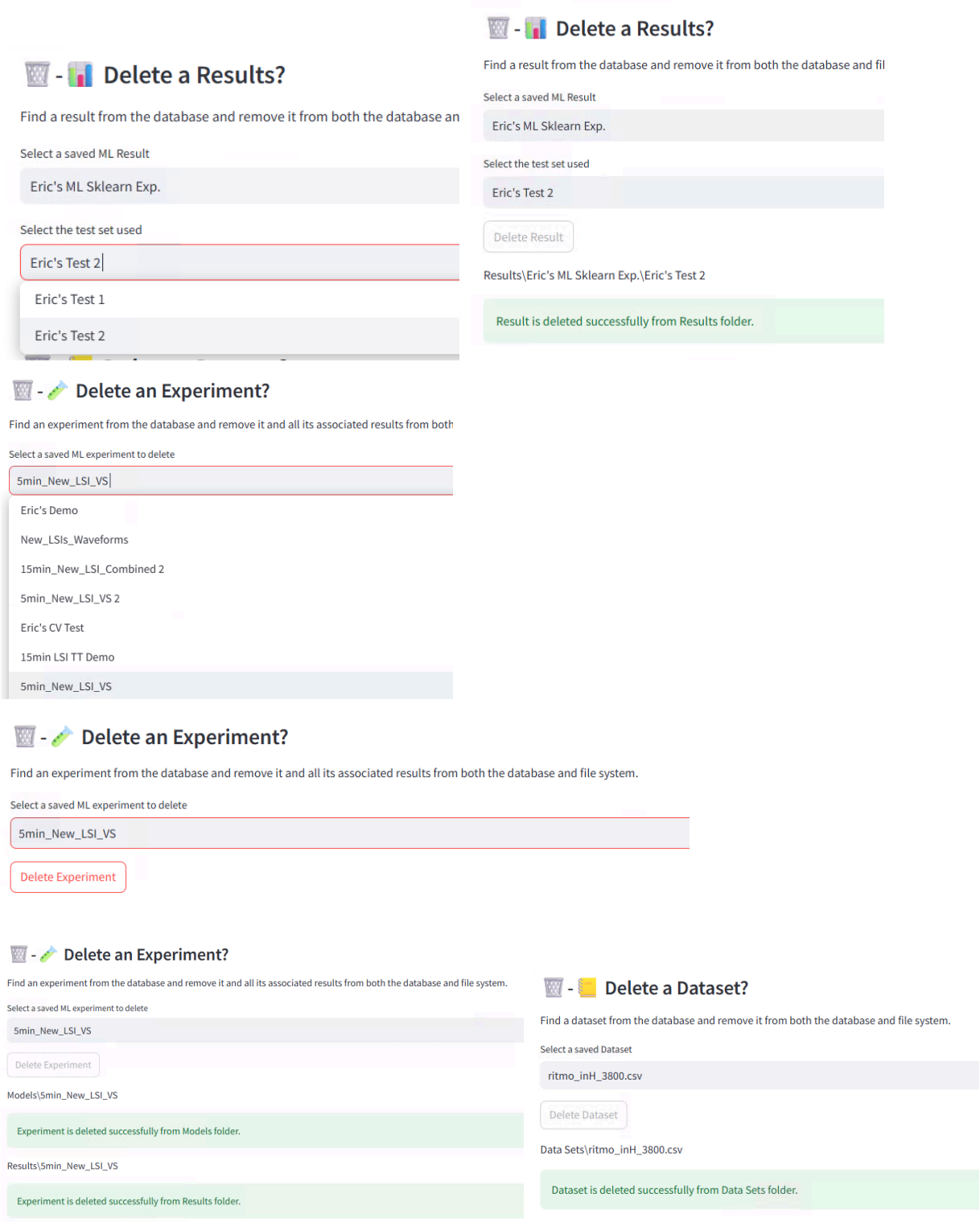
Screenshot of the management section



When the user first clicks on the management button in the main page, this is the first thing they will see. They can switch between two tabs, either for deletions or renaming. There is also a back button on the top left corner, where the user can go back to the main page.

Screenshots of the deletion tab



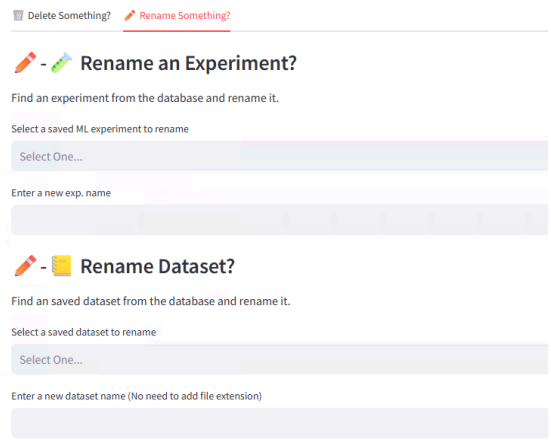


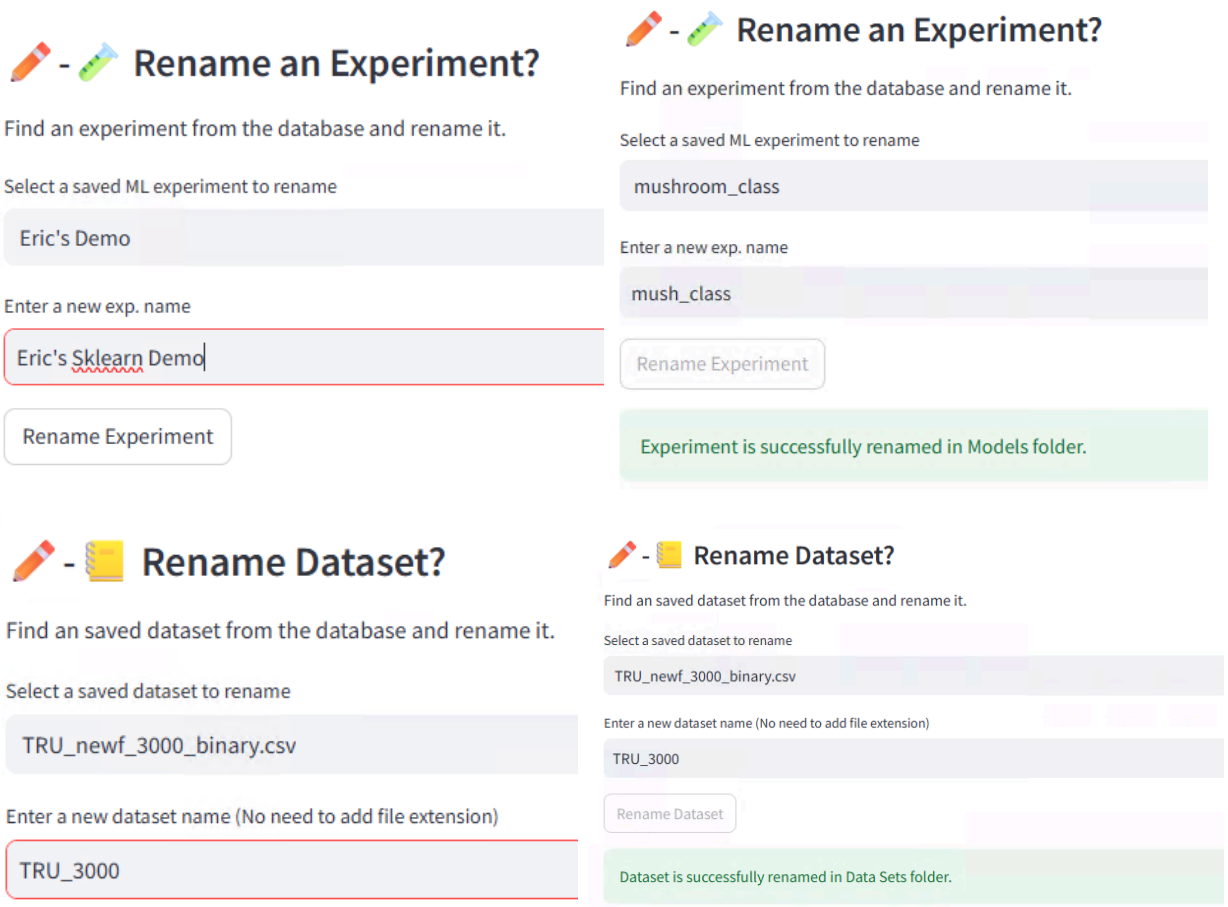
For the deletion tab, the user can use the dropdown menus to choose a specific ML experiment, a specific test from a specific ML experiment, or a specific saved dataset that they want to delete. The user then must click on the delete button below to perform the deletion of the specified item. Once it's complete, the app will tell the user if the experiment and its associated

results are successfully deleted with a green tab. If there is a red tab, that means that the item (mostly happens with test results of an experiment) is not available in the database, thus can’t be deleted.

5.7 Rename

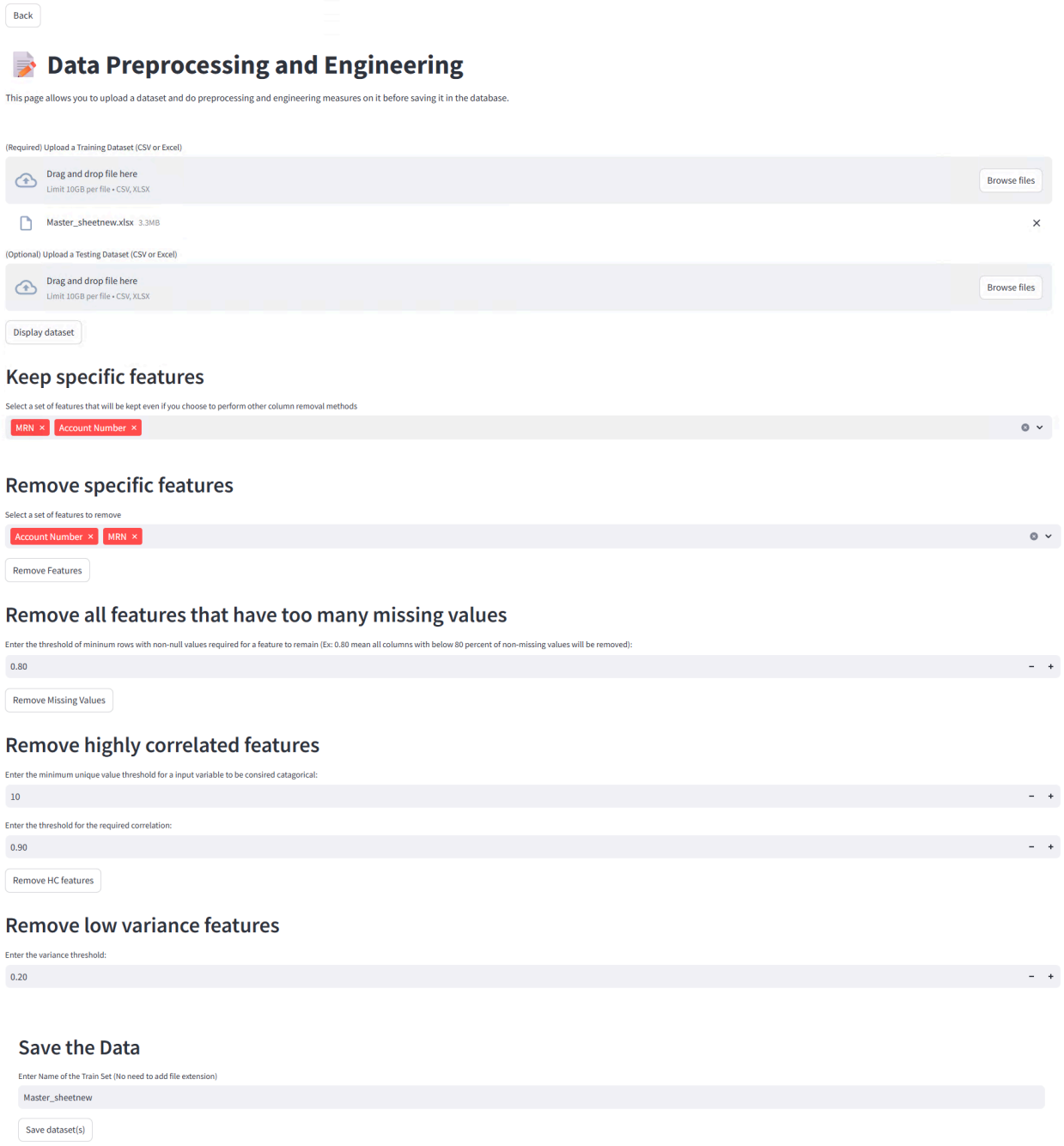
Screenshots of the rename tab





In the rename section, the user can also use the presented dropdown menus there to obtain a specific ML experiment or dataset and then use the textbox below to give it a new name. The user then must click on the rename button below to perform the renaming action, in which then the app will tell the user if the renaming action is successful.

5.8 Data Preprocessing



In the data preprocessing section, the user can upload a dataset and then do various preprocessing and engineering steps on the dataset. This includes removing specific features, removing highly correlated or low variance features, and removing features that have too many missing values. After that, the user can save the data into the dataset to be used later in the training section. The user can also specify which columns to keep in which they are protected from many data removals.