## Files

In the project directory under ./datastores/csv, there will be a zipped folder called “datafiles.zip”. Please unzip these. There should be 2 files called “sample_10000_train.csv” and “sample_2000_test.csv”. These will be files referenced for the training and testing step, respectively.

## Usage

### Local Installation

In order to run the Midas pipeline locally, first install Docker and docker-compose.
Then, in terminal type the following commands:

```
docker-compose build
docker-compose up
```

Once the commands complete, use a browser and navigate to “localhost”.
If, for some reason, the process does not complete successfully, another attempt with the docker commands may be required.

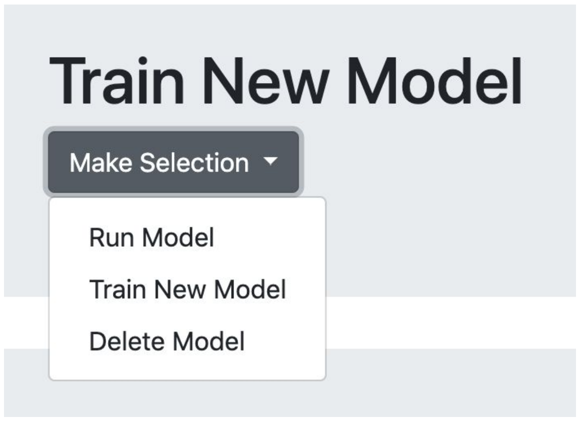
### AWS

Access the AWS instance of the pipeline at the following url:
http://ec2-18-219-155-220.us-east-2.compute.amazonaws.com/#

### Pipeline Walkthrough

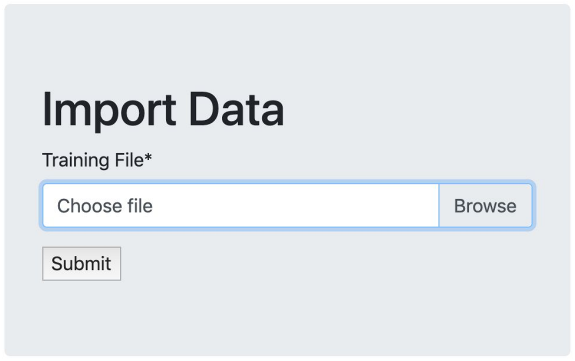
#### Task Selection

The first step for the user is to select the type of task. For a first time user, the first task will be to “Train New Model”, which trains a new machine learning model using training data. Once at least one model has been generated and saved, the user may want to use a model to make predictions on new data, or may wish to delete a saved model. Most of this walkthrough focuses on the process of training a new model since this is the most complicated task.



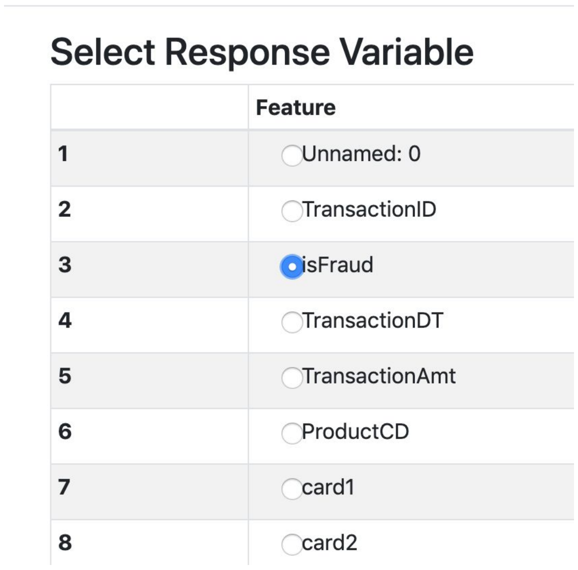
## Data Import

In the data import step, the user browses and selects a training dataset to import (csv required). Once the user submits the file selection, the file is then uploaded. Please import the sample_10000_train.csv file.



## Data Preparation

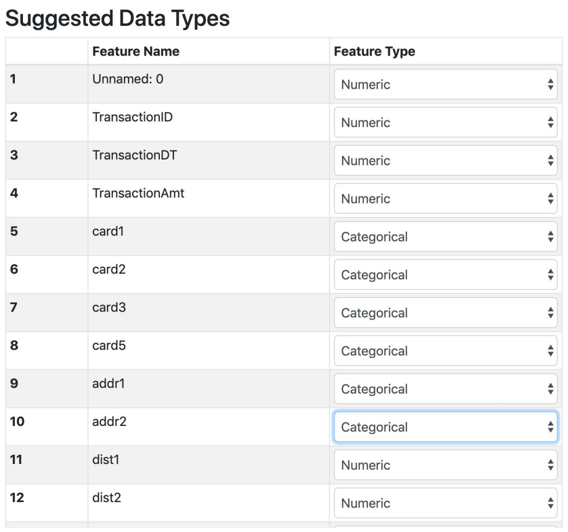
The first step in the data preparation component is to choose the response variable.  In our example fraud detection dataset, the outcome variable is called “isFraud”.  Then, scroll to the bottom of the page and click “Submit”.



Select Response Variable

| | Feature |
|---|---|
| 1 | ○ Unnamed: 0 |
| 2 | ○ TransactionID |
| 3 | ● isFraud |
| 4 | ○ TransactionDT |
| 5 | ○ TransactionAmt |
| 6 | ○ ProductCD |
| 7 | ○ card1 |
| 8 | ○ card2 |

The second step of the data preparation task is to confirm the data types. Our pipeline infers data types automatically based on whether the field contains only numeric data (numeric) or some text data (categorical). However, sometimes variables only contain numeric data but are, in fact, categorical. In this particular example dataset, features “addr1”, “addr2” and “card1” through “card6” are all categorical (“card4” and “card6” are already correctly identified as categorical. These can be changed by clicking “Confirm Data Type” and finding these features in the “Suggested Data Types” page.



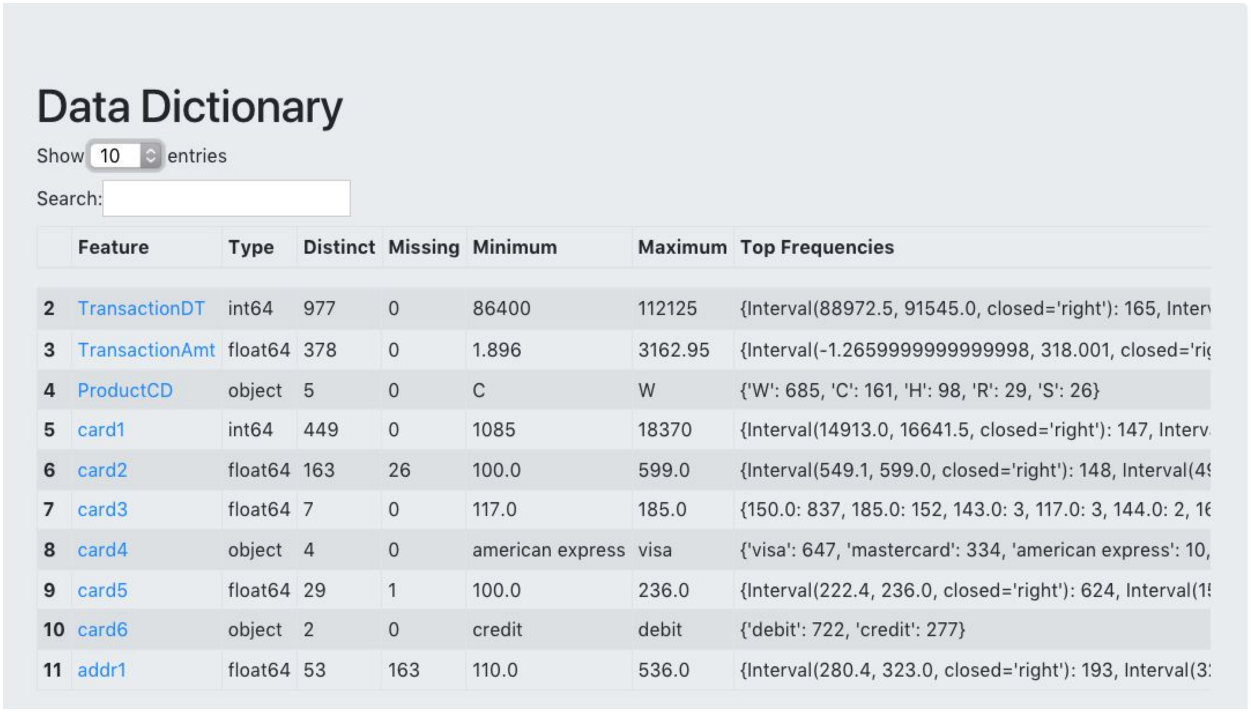
Suggested Data Types

| | Feature Name | Feature Type |
|---|---|---|
| 1 | Unnamed: 0 | Numeric |
| 2 | TransactionID | Numeric |
| 3 | TransactionDT | Numeric |
| 4 | TransactionAmt | Numeric |
| 5 | card1 | Categorical |
| 6 | card2 | Categorical |
| 7 | card3 | Categorical |
| 8 | card5 | Categorical |
| 9 | addr1 | Categorical |
| 10 | addr2 | Categorical |
| 11 | dist1 | Numeric |
| 12 | dist2 | Numeric |

## Data Analysis

After data import, the data are analyzed. Our software will automatically collect summary and detailed statistics about each feature. We added a search bar to speed inspecting specific features.

The summary statistics will show for each feature the data type, the number of distinct values, the number of missing values, the minimum value, maximum value, and a breakdown of the frequencies grouped into intervals.



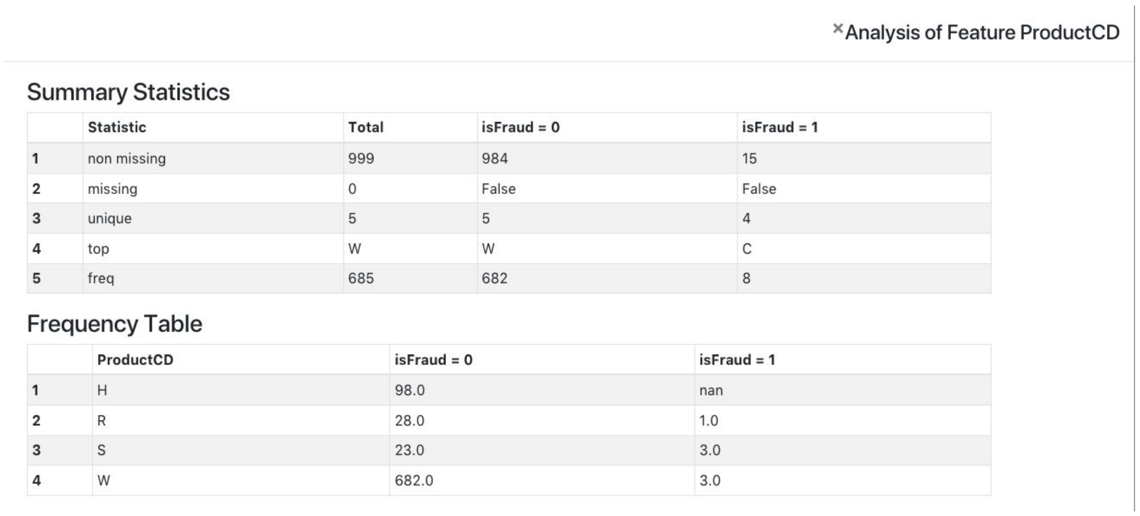
## Data Dictionary

Show 10 entries

Search:

| | Feature | Type | Distinct | Missing | Minimum | Maximum | Top Frequencies |
|---|---|---|---|---|---|---|---|
| 2 | TransactionDT | int64 | 977 | 0 | 86400 | 112125 | {Interval(88972.5, 91545.0, closed='right'): 165, Inter |
| 3 | TransactionAmt | float64 | 378 | 0 | 1.896 | 3162.95 | {Interval(-1.2659999999999998, 318.001, closed='ri |
| 4 | ProductCD | object | 5 | 0 | C | W | {'W': 685, 'C': 161, 'H': 98, 'R': 29, 'S': 26} |
| 5 | card1 | int64 | 449 | 0 | 1085 | 18370 | {Interval(14913.0, 16641.5, closed='right'): 147, Interv |
| 6 | card2 | float64 | 163 | 26 | 100.0 | 599.0 | {Interval(549.1, 599.0, closed='right'): 148, Interval(4 |
| 7 | card3 | float64 | 7 | 0 | 117.0 | 185.0 | {150.0: 837, 185.0: 152, 143.0: 3, 117.0: 3, 144.0: 2, 1 |
| 8 | card4 | object | 4 | 0 | american express | visa | {'visa': 647, 'mastercard': 334, 'american express': 10, |
| 9 | card5 | float64 | 29 | 1 | 100.0 | 236.0 | {Interval(222.4, 236.0, closed='right'): 624, Interval(1 |
| 10 | card6 | object | 2 | 0 | credit | debit | {'debit': 722, 'credit': 277} |
| 11 | addr1 | float64 | 53 | 163 | 110.0 | 536.0 | {Interval(280.4, 323.0, closed='right'): 193, Interval(3 |

The user may click on a feature like ProductCD to get detailed statistics that show summary statistics, frequency statistics, and a bar chart or histogram (depending on whether the data are categorical or numerical) and how these values vary between outcome classes. Specifically, examples of the summary statistics include counts of non-missing, missing, unique values, and the top most frequent (mode). The frequency table shows lists the unique values of the feature and counts of observations for each (by outcome variable value). An example is shown below:



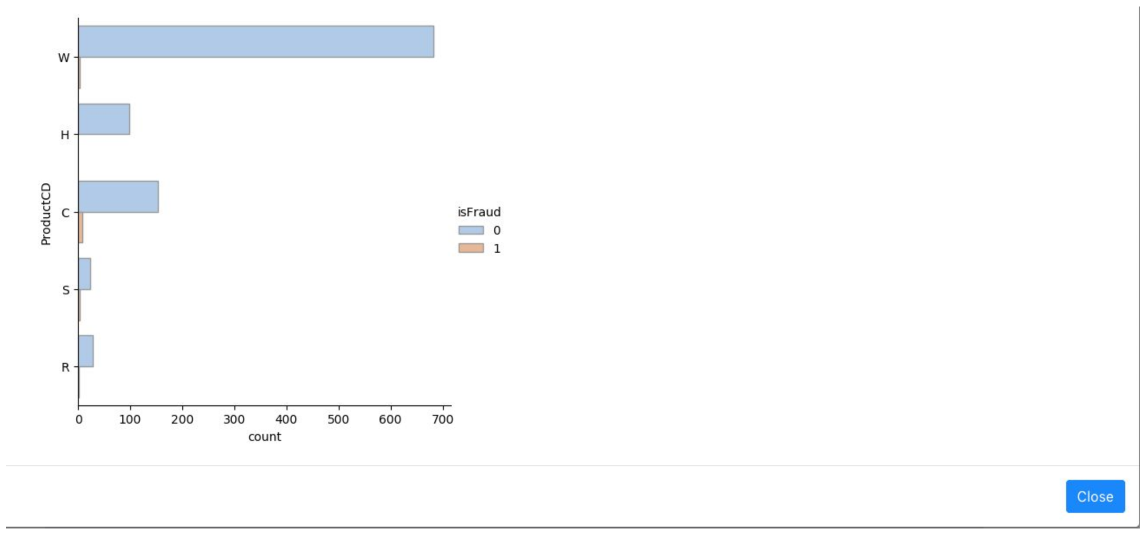
×Analysis of Feature ProductCD

### Summary Statistics

| | Statistic | Total | isFraud = 0 | isFraud = 1 |
|---|---|---|---|---|
| 1 | non missing | 999 | 984 | 15 |
| 2 | missing | 0 | False | False |
| 3 | unique | 5 | 5 | 4 |
| 4 | top | W | W | C |
| 5 | freq | 685 | 682 | 8 |

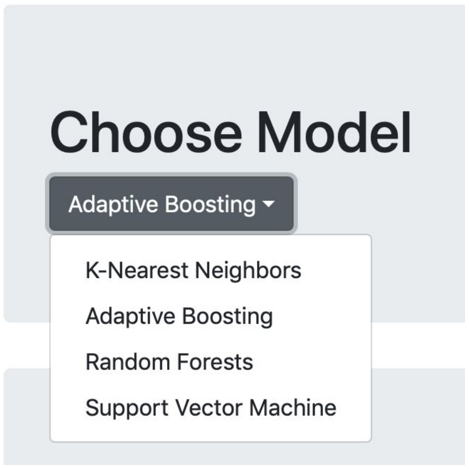
### Frequency Table

| | ProductCD | isFraud = 0 | isFraud = 1 |
|---|---|---|---|
| 1 | H | 98.0 | nan |
| 2 | R | 28.0 | 1.0 |
| 3 | S | 23.0 | 3.0 |
| 4 | W | 682.0 | 3.0 |



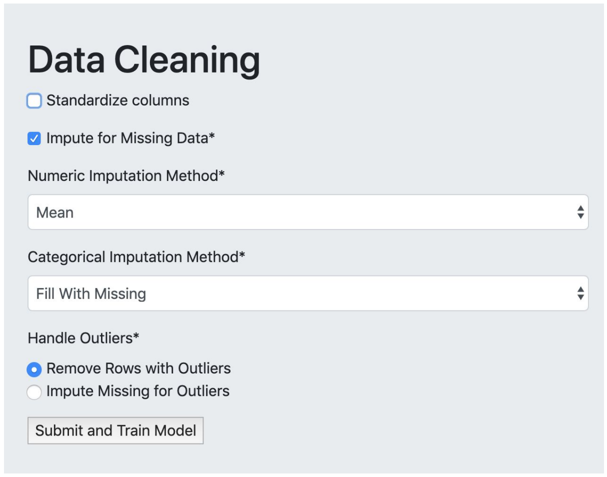
## Algorithm Selection

In the next step, the machine learning algorithm is selected. This is done before data cleaning because the relevant data cleaning steps vary based on the machine learning algorithm. There are four options presented: K-Nearest Neighbors, AdaBoost, Random Forests, and Support Vector Machines. K-Nearest Neighbors is an algorithm that makes predictions based on the K nearest neighbors (observations) in the feature space. Our implementation uses 10-fold cross validation to select the best value of K. AdaBoost and Random Forests are both ensemble tree-based algorithms for classification. AdaBoost uses an iterative approach incrementally improving the model over multiple stages. Our pipeline uses cross validation to select the best learning rate for AdaBoost. Random Forests uses multiple decision trees and aggregates the results for the prediction. Our pipeline uses cross validation to select the best number of classifiers (decision trees). Support Vector Machine find a decision boundary that best separates the feature space between the two outcome classes. For SVM, we use cross validation to select the best kernel (linear, polynomial, or radial basis function).



## Data Cleaning and Training

In the data cleaning and training stage, users make choices about appropriate data cleaning steps. Different data cleaning steps are required for different algorithms (is missing data tolerated, is standardization of numeric features recommend, etc.). The user is offered different methods for imputing missing data and how to handle outliers. Once the user's selections are made, click on "Submit and Train Model" to clean the data and then train the learning machine learning algorithm based on the specifications. Our machine is not optimized for training large datasets, and thus it is recommended to select relatively small datasets.



Data Cleaning

☐ Standardize columns

☑ Impute for Missing Data*

Numeric Imputation Method*: Mean

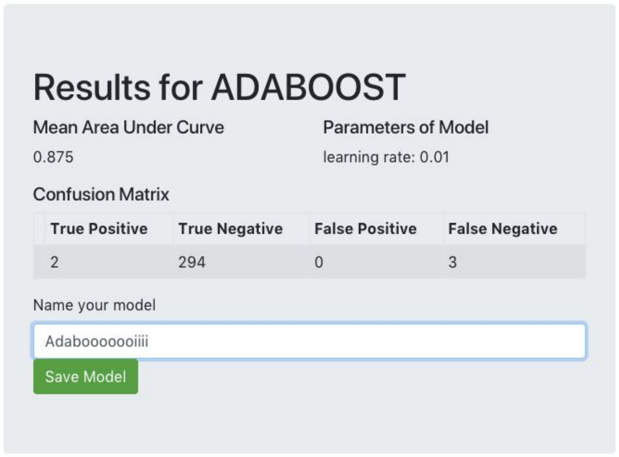
Categorical Imputation Method*: Fill With Missing

Handle Outliers*

● Remove Rows with Outliers
○ Impute Missing for Outliers

## Summary of Results

After training is finished, we populate the results with a table showing the mean area under the curve, the parameters used in the machine learning algorithm, and a confusion matrix (count of true positives, false positives, true negatives and false negatives). For reference, the ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The area under the ROC curve is a metric that captures how well the model is able to distinguish between the outcome classes. The values range from 0 to 1. A value close to 1 indicates that the model is able to predict the classes very well. A value closer to 0.5 indicates that the model is not that effective, close to predicting at random. The user may then choose to give a name to the model to save it for later execution.



Results for ADABOOST

Mean Area Under Curve

0.875

Parameters of Model

learning rate: 0.01

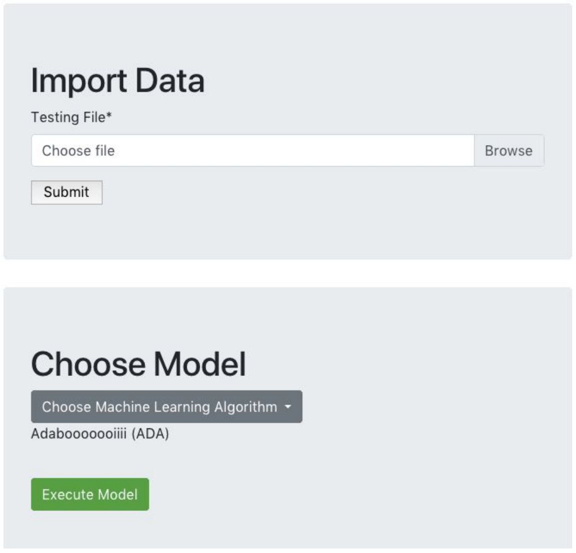
Confusion Matrix

| True Positive | True Negative | False Positive | False Negative |
|---|---|---|---|
| 2 | 294 | 0 | 3 |

Name your model: Adabooooooiiii

## Importing New Data and Executing Models

When the user finishes training new models, they may then switch to running models against new data that they import. The user imports the new test file which should have the same columns as the training data but without the labels. Please import the sample_2000_test.csv file.



Import Data

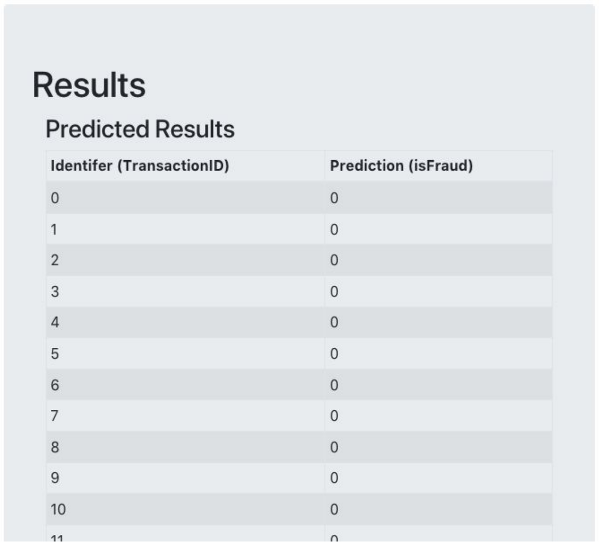
Testing File*

Choose Model

Adabooooooiiii (ADA)

## Results of Executing the Model

The page then shows the results of executing the model. We have not implemented a way of storing or returning these results back to the user, so as of now the results can only be viewed in the browser.



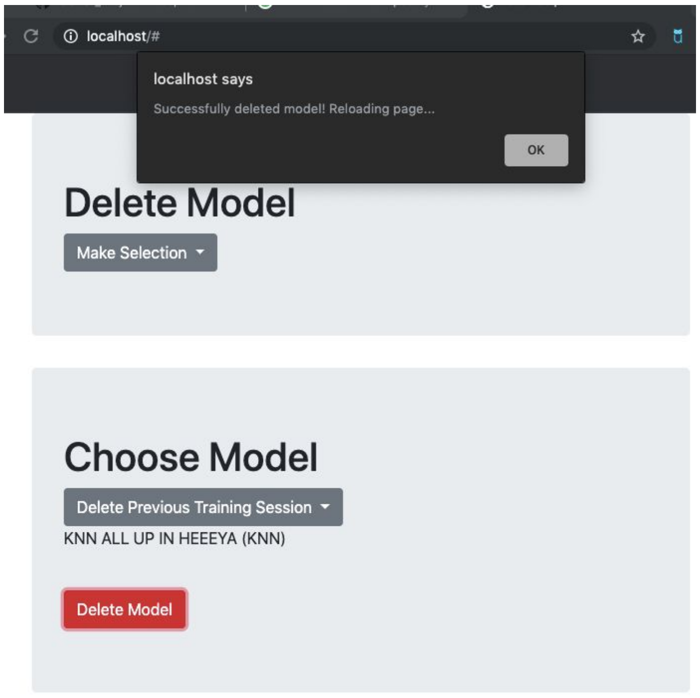
Results

Predicted Results

| Identifer (TransactionID) | Prediction (isFraud) |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |

## Deleting Past Models

The user may also choose to delete old models



localhost/#

localhost says

Successfully deleted model! Reloading page...

KNN ALL UP IN HEEEYA (KNN)