

 README.md

Accelerate data preparation using Amazon SageMaker Data Wrangler for Diabetic Patient Readmission Prediction

Patient readmission to hospital after prior visits for the same disease results in additional burden on healthcare providers, health system and patients. Machine Learning (ML) models if built and trained properly can help understand reasons for readmission, and predict readmission accurately. ML could allow providers to create a better treatment plans and care which would translate to reduction of both cost and mental stress for patients. However, ML is a complex technique that has been limiting organizations that do not have the luxury to recruit a team of data engineers and scientists to build ML workloads. In this example, we show you how to build a machine learning model to predict diabetic patient readmission easily and quickly with a graphical interface from Amazon SageMaker Data Wrangler.

Amazon SageMaker Data Wrangler is an Amazon SageMaker Studio feature designed to allow users to explore and transform tabular data for machine learning use cases without coding. Amazon SageMaker Data Wrangler is the fastest and easiest way to prepare data for Machine Learning. It gives you the ability to use a visual interface to access data, perform exploratory data analysis (EDA) and feature engineering. It also seamlessly operationalizes your data preparation steps by allowing to export data flow into Amazon SageMaker Pipelines, Amazon SageMaker Data Wrangler job, Python file or Amazon SageMaker Feature Store.

Amazon SageMaker Data Wrangler comes with over 300 built-in transforms, custom transformations using either Python, PySpark or SparkSQL runtime. It also comes with built-in data analysis capabilities for charts (eg, scatterplot or histogram) and time-saving model analysis capabilities such as Feature importance, Target leakage and Model explainability.

In this step-by-step example, you will be running machine learning workflow with Amazon SageMaker Data Wrangler and Amazon SageMaker features using a HCLS dataset.

Here are the high-level activities:

1. Load UCI Source Dataset into your S3 bucket
2. Design your DataWrangler flow file
3. Processing & Training Jobs for Model building
4. Host trained Model for real-time inference

1. Source Dataset

UCI diabetic patient readmission dataset. The dataset represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. It includes over 50 features representing patient and hospital outcomes.

You will start by downloading the dataset and uploading it to a S3 bucket for you to run the example. Please review and execute the code in `datawrangler_workshop_pre_requisite.ipynb`. The data will be available in

`s3://sagemaker-${region}-${account_number}/sagemaker/demo-diabetic-datawrangler/` if you leave everything default.

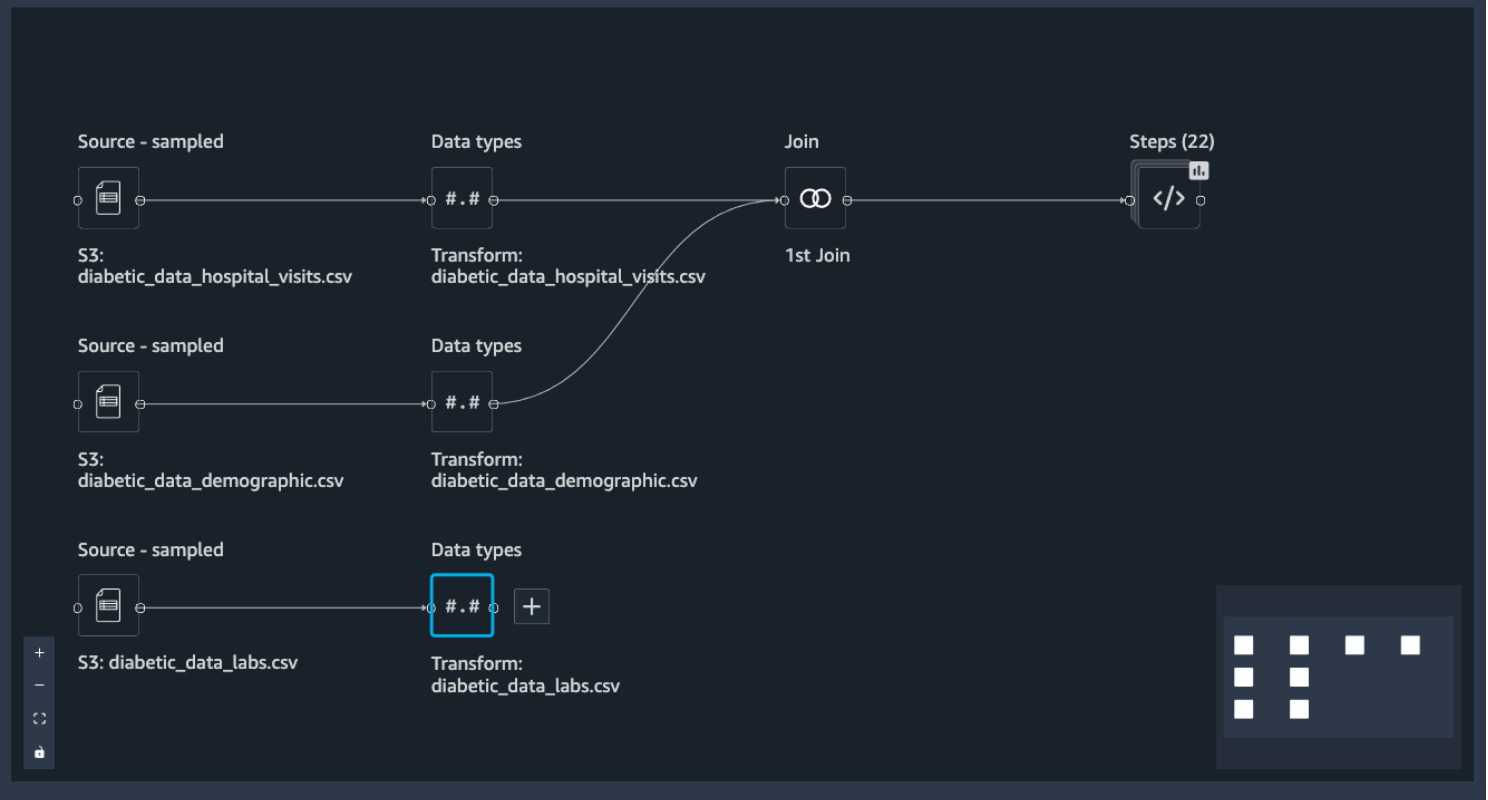
2. Design your DataWrangler flow file

Data Wrangler flow overview and highlights

This project comes with a pre-built Data Wrangler flow file that can be customized with your `s3Uri` for reusability: `datawrangler_diabetes_readmission.flow`.

Data flow

Choose the plus sign to add a step to the flow. Select a step to modify.



It has multiple files from S3 loaded in: `diabetic_data_hospital_visits.csv` , `diabetic_data_demographic.csv` and `diabetic_data_labs.csv` for demonstration. It performs an inner join between the tables in `diabetic_data_hospital_visits.csv` and `diabetic_data_demographic.csv` by `encounter_id` . It has 28 transformation steps applied to process the data to meet the following requirements:

- no duplicate columns
- no duplicate entries
- no missing values (either fill the missing ones or remove columns that are largely missing)
- one hot encode the categorical features
- ordinal encode the age feature
- normalization (Standard scaler)
- Custom Transformation (Feature Store - EventTime needed)
- Analysis (Quick Model, Histogram)
- ready for ML training (Export notebook steps)

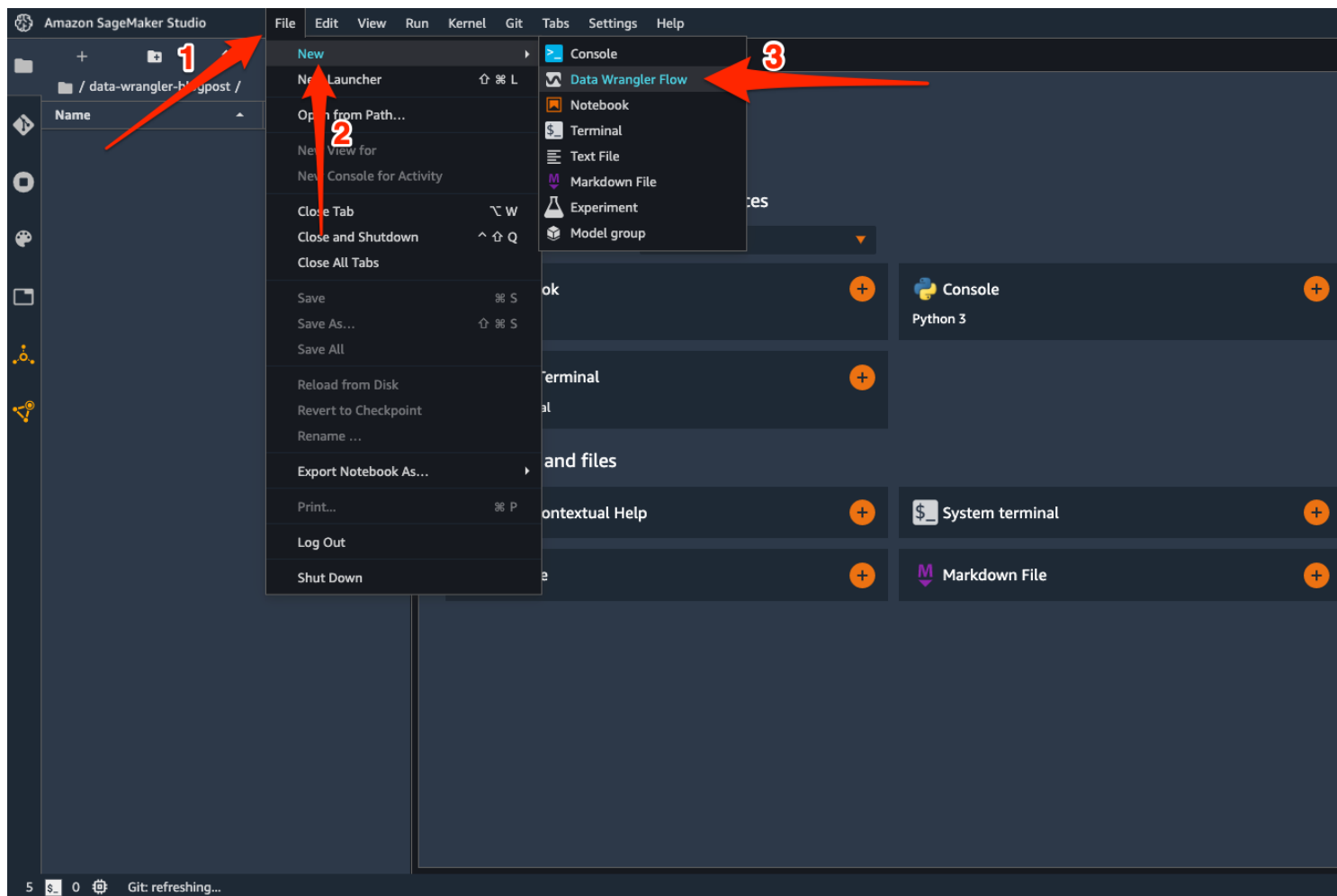
These are analyses created at different stage of the wrangling to serve as indication of the value these wrangling steps add. Most noticeably the Quick Model tells us that patient readmission prediction increases F1 score after performing transformation steps between 1 and 28 (in `datawrangler_diabetes_readmission.flow`). Data Scientists can use `Quick Model` analysis to perform iterative experimentation leading to efficient feature engineering for ML.

In this lab, we will perform data preprocessing using a combination of transformations described below to demonstrate the capability of Amazon SageMaker Data Wrangler. We will then train a XGBoost model to show you the process after data wrangling. We will then be hosting a trained model to SageMaker Hosted Endpoint for real-time inferencing.

Walk through

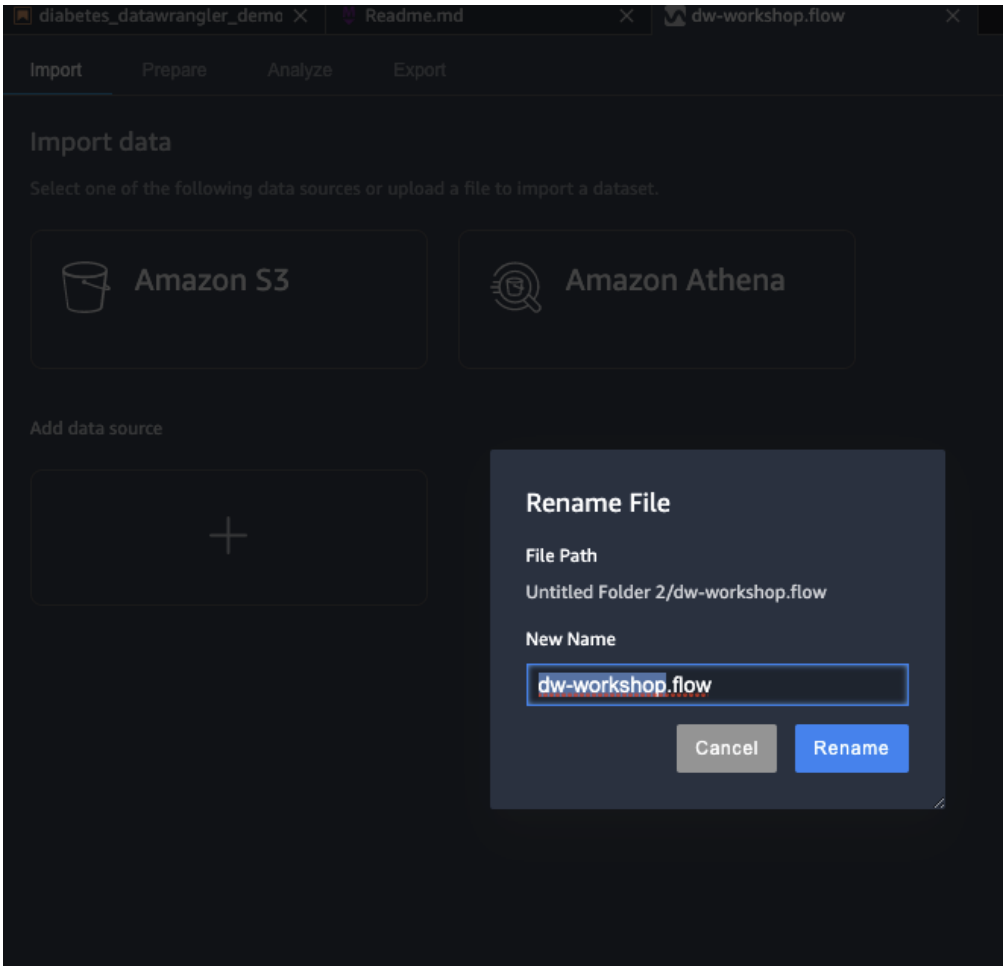
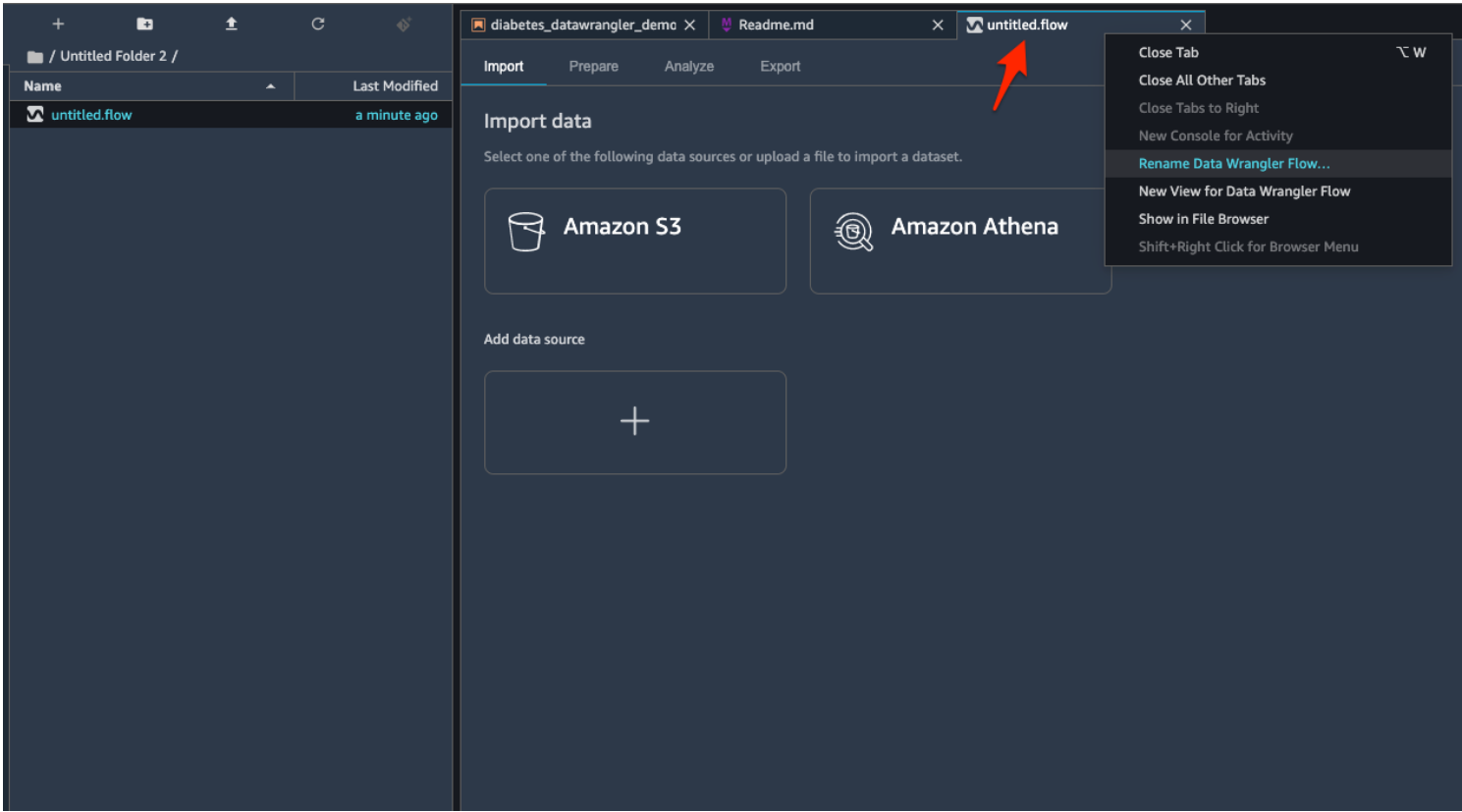
Create a new flow

Please click on the **SageMaker component and registry tab** and click **New flow**.



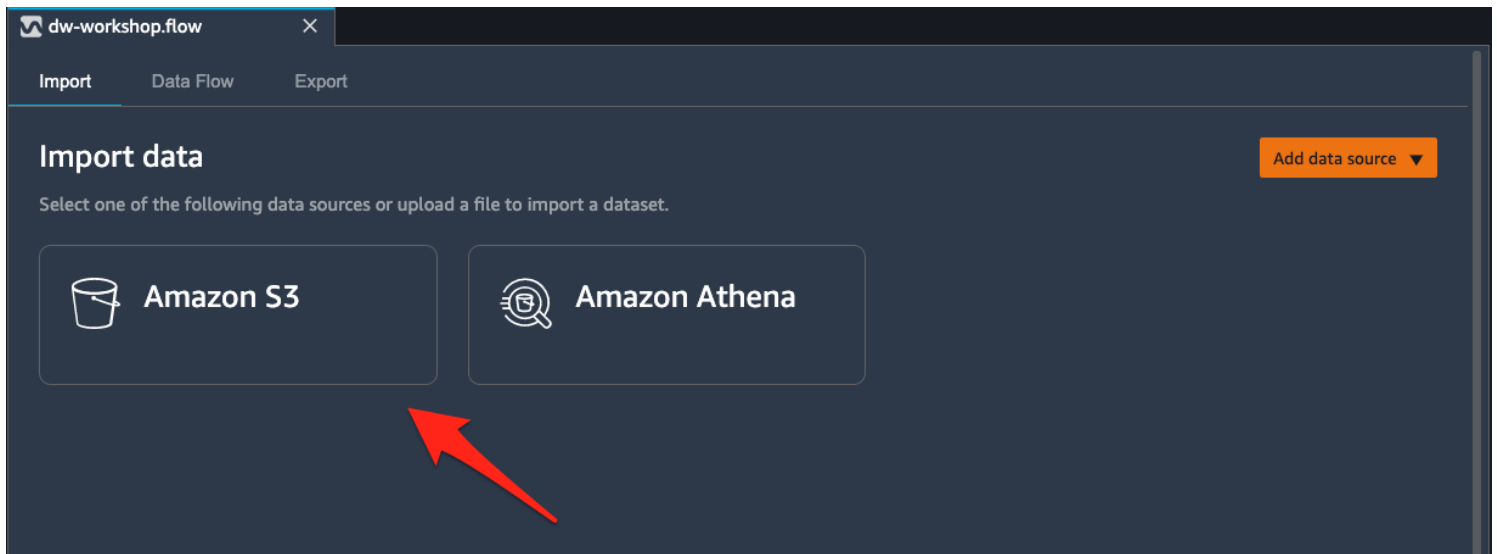
Rename your DW dataflow file

Right click on the **untitled.flow** file tab to reveal below options. Then choose **Rename file** to change the file name.



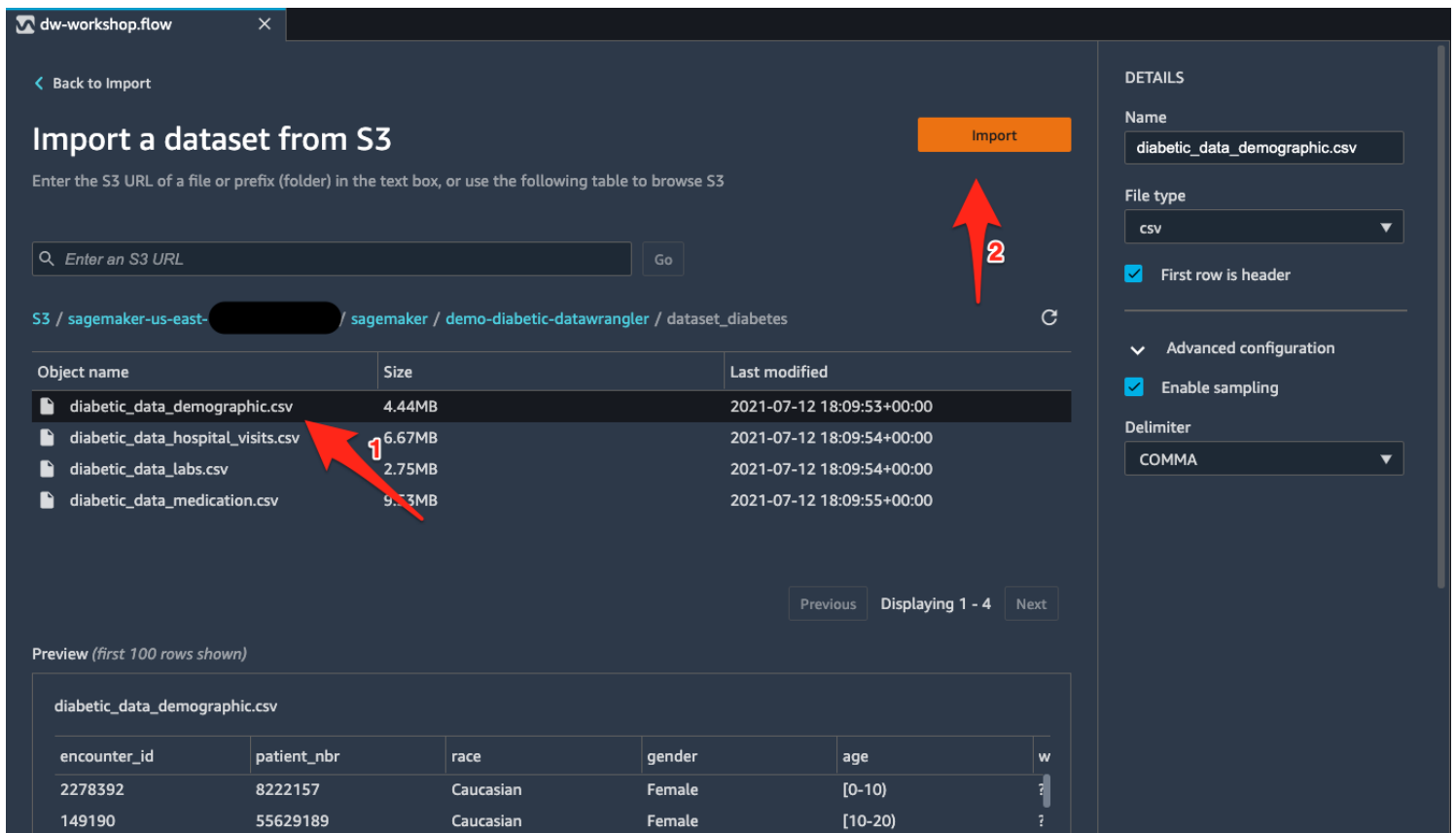
Load the data from S3 into Data Wrangler

Select Amazon S3 as data source in **Import Data** view.



Note: You could also import data from Athena: how databases and tables in Amazon Athena can be imported.

Select the csv files from the bucket: `s3://sagemaker-${region}-${account_number}/sagemaker/demo-diabetic-datawrangler/` one at a time.



Join the CSV files

1. Click the + sign on the Data-types icon for `diabetic_data_demographic.csv`. Select **Join** and new panel is presented for configuring input dataset join.

dw-workshop.flow

Import Data Flow Export

Data flow

Choose the plus sign to add a step to the flow. Select a step to modify.

The screenshot shows the Grip Data Flow interface. It has a dark theme with a top navigation bar containing 'Import', 'Data Flow' (selected), and 'Export'. Below the navigation bar is a header 'Data flow' with a sub-header 'Choose the plus sign to add a step to the flow. Select a step to modify.' The main workspace displays two data sources on the left, each with a 'Source' icon and a green checkmark. The first source is 'S3: diabetic_data_demographic.csv' and the second is 'S3: diabetic_data_hospital_visits.csv'. Both sources are connected to a 'Data types' step on the right, which has a green checkmark. The 'Data types' step for the first source is labeled 'Transform: diabetic_da' and has a plus sign icon. A red arrow labeled '1' points to this plus sign. The 'Data types' step for the second source is labeled 'Transform: diabetic_data_hospital_visits.csv' and has a plus sign icon. A red arrow labeled '2' points to this plus sign. A context menu is open over the plus sign of the second 'Data types' step, showing options: 'Edit data types', 'Add transform', 'Add analysis', 'Join' (highlighted in blue), and 'Concatenate'. In the bottom right corner, there is a small preview window showing a grid of four squares, with the bottom-right square highlighted in blue.

2. Select `diabetic_data_hospital_visits.csv` dataset as Right dataset.

3. Click `Configure` to setup Join criteria.

Data flow / Join

Join

1. Select datasets 2. Configure

Cancel Add

Datasets

- Left
Transform: diabetic_data_demographic.csv
- Right
Transform: diabetic_data_hospital_visits.csv

Select the dataset for the Right

Source ✓ Data types ✓ Transform

S3: diabetic_data_demographic.csv #.# Transform: diabetic_data_demographic.csv Join preview

S3: diabetic_data_hospital_visits.csv #.# Transform: diabetic_data_hospital_visits.csv

Back Configure

2

1

4. Give a name to the Join and choose join type and Left & Right columns for join condition

5. Click Apply to preview the joined dataset and Add for the previewed join configuration to be added to the data-flow file.

Data flow / Join

Join

1. Select datasets 2. Configure

Cancel Add

Datasets

Left
Transform: diabetic_data_demographic.csv

Right
Transform: diabetic_data_hospital_visits.csv

Joined dataset
Name
demographics_hospitalvisits_join

Optional
Join Type
Select the join type
Inner

Columns
Select Left and Right to join
Left
encounter_id
Right
encounter_id

Back Apply

Preview

INPUT

Left
Transform: diabetic_data_demographic.csv

encounter_id (long)	patient_nbr (long)
2278392	8222157
149190	55629189
64410	86047875
500364	82442376
16680	42519267
35754	82637451
55842	84259809

Right
Transform: diabetic_data_hospital_visits.csv

encounter_id (long)	patient_nbr (long)
2278392	8222157
149190	55629189
64410	86047875
500364	82442376
16680	42519267
35754	82637451
55842	84259809

OUTPUT

Joined dataset
demographics_hospitalvisits_join

encounter_id_0 (long)	patient_nbr_0 (long)	race (string)	gender (string)	age (string)
2278392	8222157	Caucasian	Female	[0-10]
149190	55629189	Caucasian	Female	[10-20]
64410	86047875	AfricanAmerican	Female	[20-30]
500364	82442376	Caucasian	Male	[30-40]

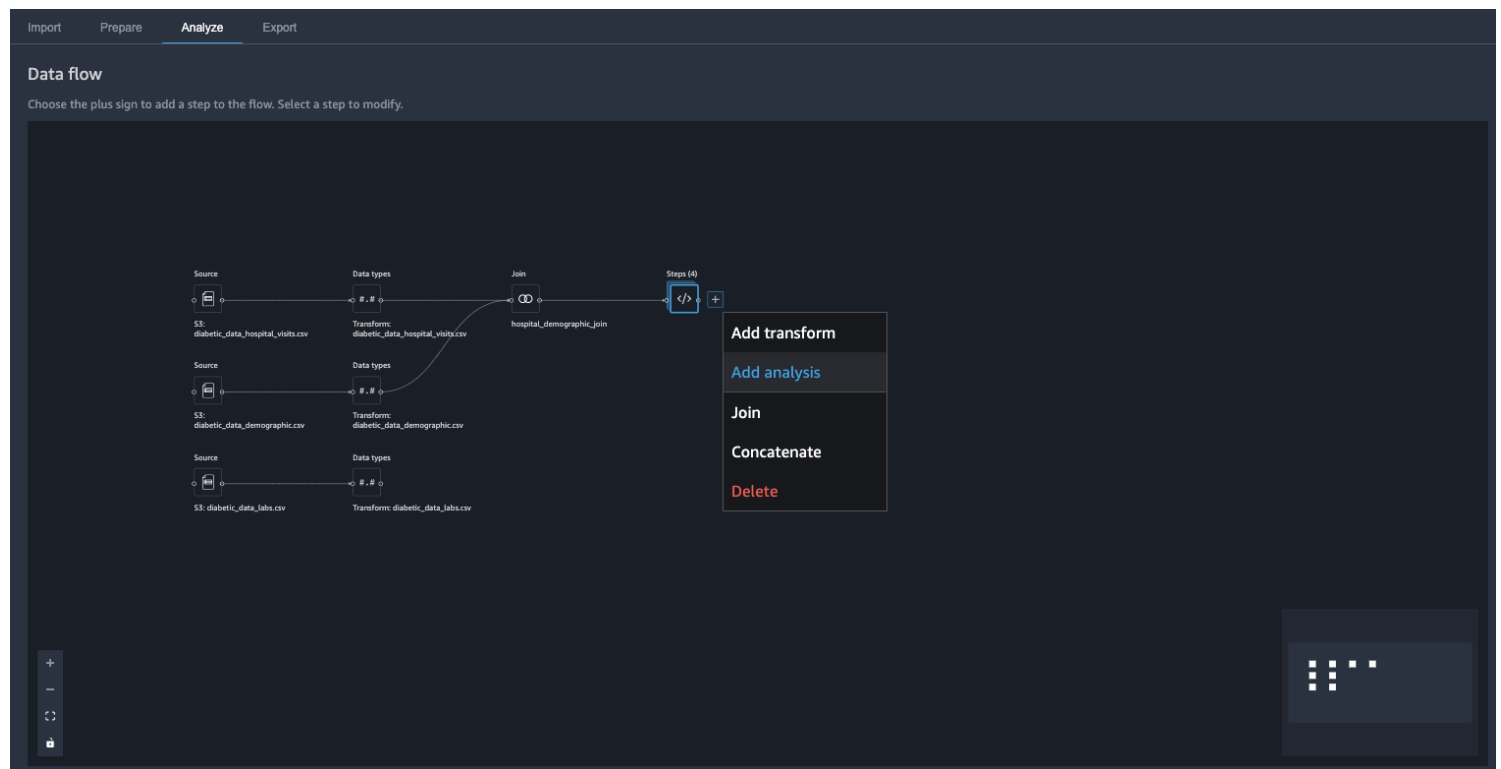
Built-in Analysis

Before we apply any transformations on the input source, let's perform a quick analysis of the dataset. SageMaker Data Wrangler provides number of built-in Analysis types like Histogram, Scatter Plot, Target Leakage, Bias Report & Quick Model. You can find all analyses types documentation under SageMaker Data Wrangler Analyses.

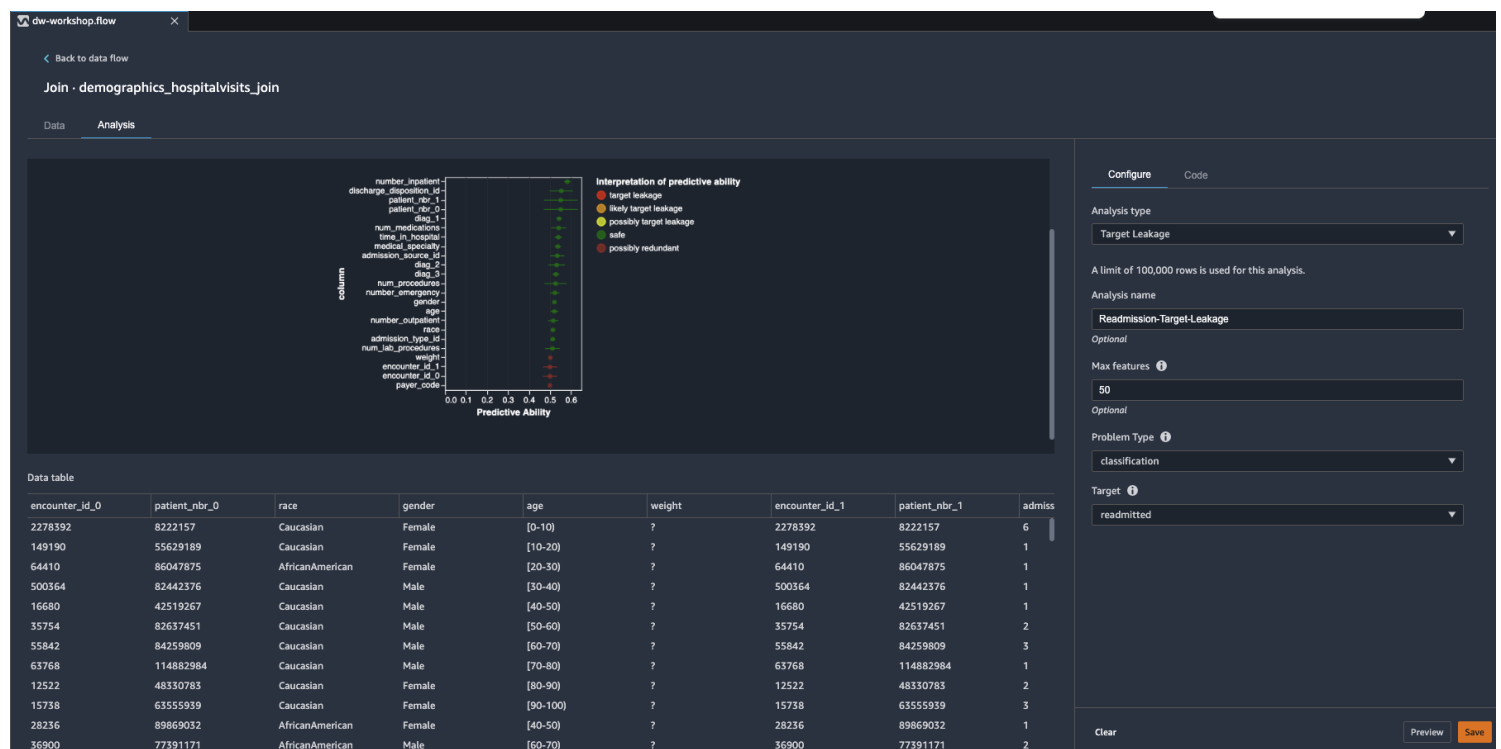
Target Leakage

Target leakage occurs when there is data in a ML training dataset that is strongly correlated with the target label, but is not available in real-world data. For example, you may have a column in your dataset that serves as a proxy for the column you want to predict with your model. Data Wrangler calculates the predictive metric of ROC which is computed individually for each column via cross validation to generate Target Leakage Report.

1. Click + sign next to Join flow icon and choose Add analysis



2. Select Target Leakage from the list of Analysis type drop down on the right panel.
3. Give a name to your analysis and specify Max features as 50, Problem Type as classification and Target as readmitted.
4. Click Preview to generate below report.



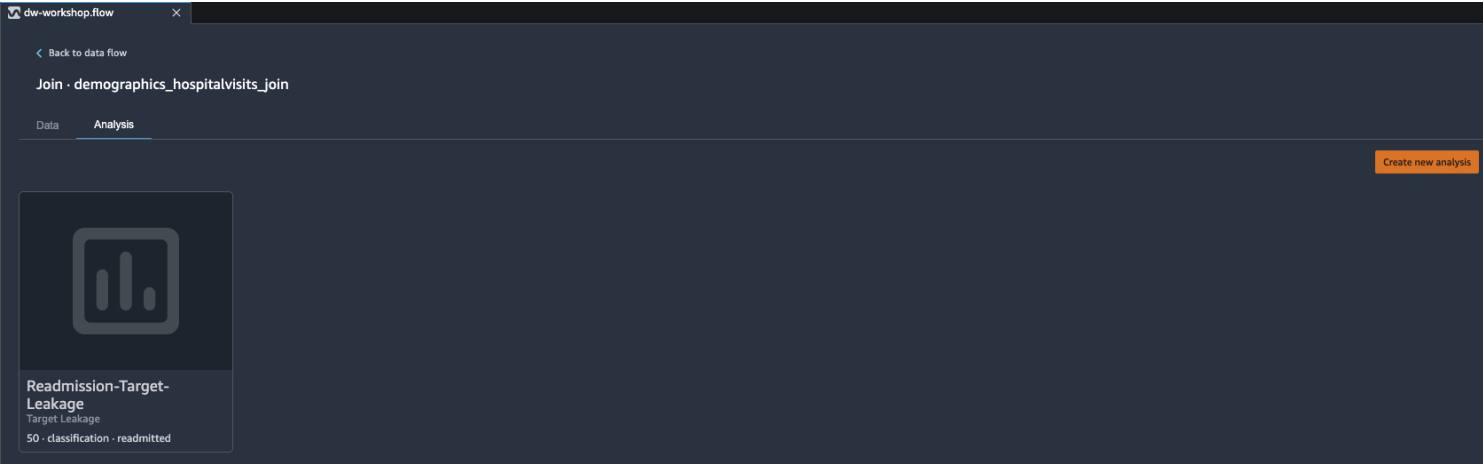
5. As shown, there is no indication of Target leakage in our input dataset. However, a few features like encounter_id_1, encounter_id_0, weight & payer_code are marked as possibly redundant with 0.5 Predictive ability of ROC. This means these features by themselves are not providing any useful information towards predicting the target. Before making the decision to drop these uninformative features, you should consider whether these could add value when used in tandem with other features. For our use-case, we'll drop these in Transforms section in an effort to prepare our training dataset.
6. Click Save to save the analysis into your Data Wrangler data flow file.

Bias Report

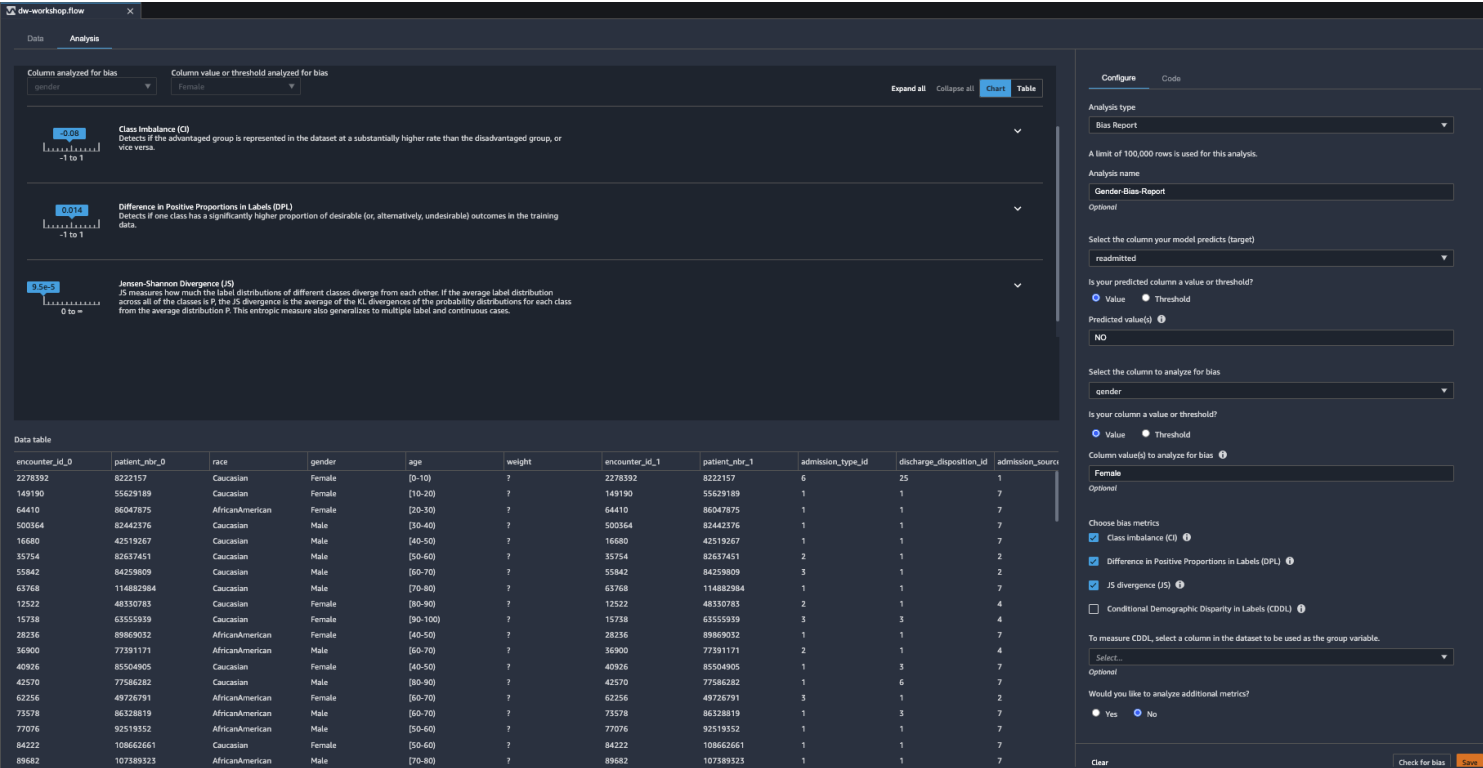
AI/ML systems are only as good as the data we put into them. ML based systems are more accessible than ever before and with the growth of adoption throughout various industries, further questions arise surrounding fairness and how it is ensured across these ML systems. Understanding how to avoid and detect bias in ML models is imperative and complex. Using Data Wrangler’s built-in Bias Report analysis, data scientists can quickly detect bias during data preparation stage of ML workflow. Bias Report analysis uses Amazon SageMaker Clarify to perform bias analysis.

To generate a bias report, you must specify the target column that you want to predict and a Facet/Column that you want to inspect for potential biases. For example, we can generate a bias report on gender feature for Female values to see whether there is any Class Imbalance (CI).

- 1. While on the Analysis tab, click `Create new analysis` button to open analysis creation panel.



- 2. Select `Bias Report` from the list of Analysis type drop down on the right panel.
- 3. Give a name to your analysis and select the target label as `readmitted` and choose Value as `NO`.
- 4. Select `gender` for column to analyze and provide value as `Female`.
- 5. Leave everything else as default and click `Check for bias` to generate the bias report.

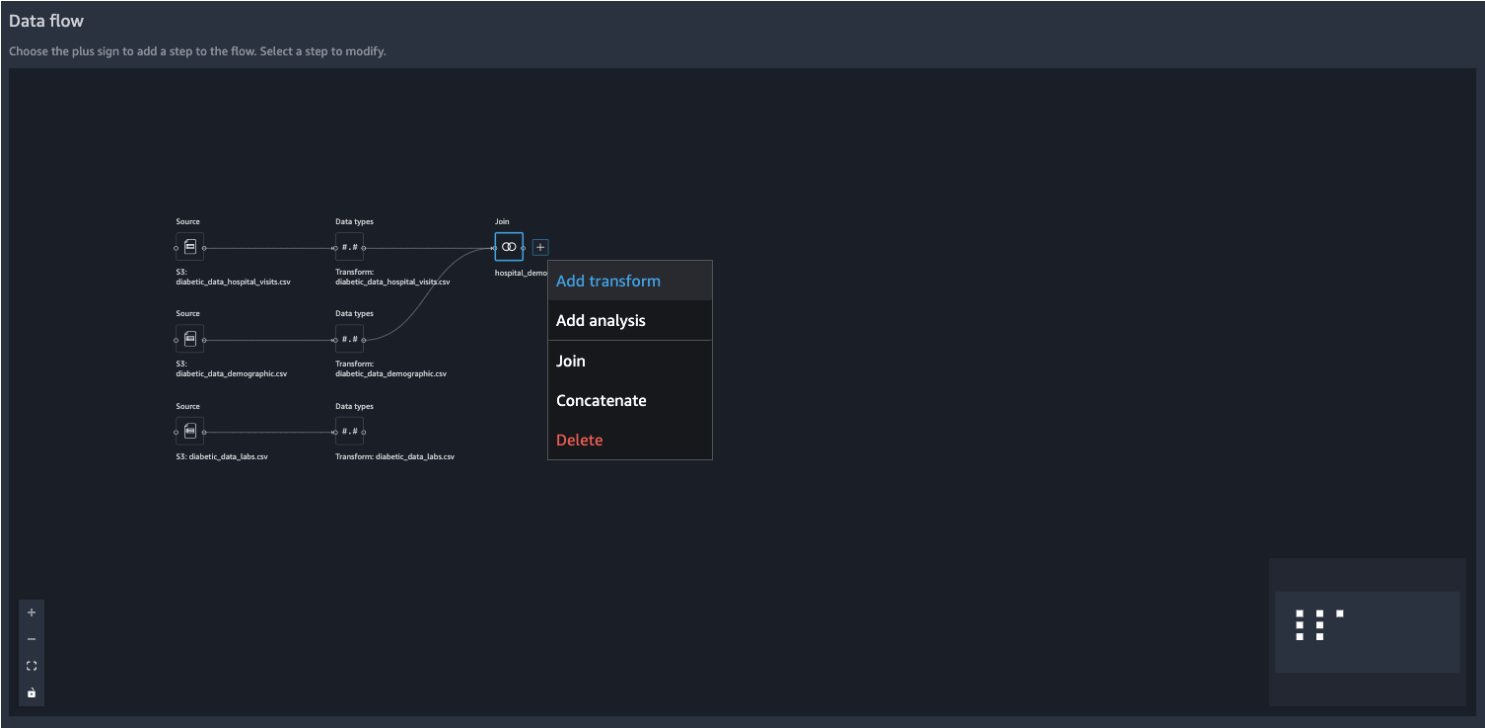


For our medical source dataset, we'll be using Amazon SageMaker built-in XGBoost algorithm as it is one of the most popular decision-tree based ensemble ML algorithm. XGBoost algorithm can only accept numerical values as input, hence a pre-requisite here is we must apply categorical feature transformations on our source dataset.

As stated, Data Wrangler comes with over 300 built-in transforms which require no coding. Let's use built-in transforms to apply a few key transformations and prepare our training dataset.

Handle missing values

- 1. Click + sign next to Join flow icon and choose Add Transform



- 2. Pick Handle missing from the list of transforms on the right panel and choose Impute for Transform

Import **Prepare** Analyze Export

Data flow / Transform: 1st Join

Transform: 1st Join

encounter_id_0 (long)	patient_nbr_0 (long)	admission_type_id (l...	discharge_dispositio...	admission_source_id ...	time_in_hospital (long)	payer_code
2278392	8222157	6	25	1	1	?
149190	55629189	1	1	7	3	?
64410	86047875	1	1	7	2	?
500364	82442376	1	1	7	2	?
16680	42519267	1	1	7	1	?
35754	82637451	2	1	2	3	?
55842	84259809	3	1	2	4	?
63768	114882984	1	1	7	5	?
12522	48330783	2	1	4	13	?
15738	63555939	3	3	4	12	?
28236	89869032	1	1	7	9	?
36900	77391171	2	1	4	7	?
40926	85504905	1	3	7	7	?
42570	77586282	1	6	7	10	?
62256	49726791	3	1	2	1	?
73578	86328819	1	3	7	12	?
77076	92519352	1	1	7	4	?
84222	108662661	1	1	7	3	?
89682	107389323	1	1	7	5	?
148530	69422211	3	6	2	6	?
150006	22864131	2	1	4	2	?
150048	21239181	2	1	4	2	?
182796	63000108	2	1	4	2	?
183930	107400762	2	6	1	11	?
216156	62718876	3	1	2	3	?
221634	21861756	1	1	7	1	?
236316	40523301	1	3	7	6	?
248916	115196778	1	1	1	2	?

TRANSFORM

Back to data flow

TRANSFORM

Add Previous steps (4)

Custom Transform

Custom formula

Encode categorical

Featurize date/time

Featurize text

Format string

Handle missing

Replace, drop, or add indicators for missing values. [Learn more.](#)

Transform

Impute

Impute

Fill missing

Add indicator for missing

Drop missing

Select...

Imputing strategy

Approximate Median

Output column

Optional

Clear Preview Add

3. Choose Column type as Numeric and select Input column as diag_1. Let's use Mean for Imputing strategy. You can also provide optional Output column name. By default, the operation is performed in-place, however, you can also provide optional Output column name which will create new column with imputed values.

4. Click Preview to preview the results as show below. Once verified, click Add to include this transformation step into Data Wrangler dataflow file.

dw-workshop.flow

Back to data flow

Previewing Handle missing

Join - demographic_hospitalvisits_join

Data Analysis

Export data

res (...)	num_procedures (long)	num_medications (long)	number_outpatient (lo...	number_emergency (l...	number_inpatient (long)	diag_1 (float)	diag_2 (long)	diag_3 (long)	readmitted (string)
0	1	0	0	0	0	250			NO
0	18	0	0	0	0	276	250	255	>30
5	13	2	0	0	1	648	250		NO
1	16	0	0	0	0	8	250	403	NO
0	8	0	0	0	0	197	157	250	NO
6	16	0	0	0	0	414	411	250	>30
1	21	0	0	0	0	414	411		NO
0	12	0	0	0	0	428	492	250	>30
2	28	0	0	0	0	398	427	38	NO
3	18	0	0	0	0	434	198	486	NO
2	17	0	0	0	0	250	403	996	>30
0	11	0	0	0	0	157	288	197	<30
0	15	0	1	0	0	428	250	250	<30
1	31	0	0	0	0	428	411	427	NO
5	2	0	0	0	0	518	998	627	>30
5	13	0	0	0	0	999	507	996	NO
4	17	0	0	0	0	410	411	414	<30
0	11	0	0	0	0	682	174	250	NO
5	23	0	0	0	0	402	425	416	>30
2	23	0	0	0	0	737	427	714	NO
1	19	0	0	0	0	410	427	428	NO
2	11	0	0	0	0	572	456	427	NO
0	12	0	0	0	0	410	401	582	NO
2	19	0	0	0	0	488.94228256609034	715		>30
4	18	0	0	0	0	189	496	427	NO
0	7	0	0	0	0	786	401	250	NO
3	18	0	0	0	0	427	428	414	NO
2	11	0	0	0	0	996	586	250	>30

TRANSFORM

Add Previous steps

Custom Transform

Custom formula

Encode categorical

Featurize date/time

Featurize text

Format string

Group by

Handle missing

Replace, drop, or add indicators for missing values. [Learn more.](#)

Transform

Impute

Column type

Numeric

The name of the column that will be created to contain the transformed data. If not set it will override the input column.

Input column

diag_1

Imputing strategy

Mean

Output column

Optional

Clear Preview Add

5. Repeat above steps 1 through 3 for diag_2 & diag_3 features and impute missing values.

Search and edit features with special characters

As our source dataset has features with special characters, we need to clean them before training. Let's use Search and Edit Transform.

1. Pick `Search and edit` from the list of transforms on the right panel. Select `Find` and `replace substring`
2. Select the target column `race` for Input column and use `\?` regex for Pattern. For the `Replacement String` use `Other` . Let's leave `Output Column` blank for in-place replacements.

Previewing Search and edit

Join - demographic_hospitalvisits_join

Data

Analysis

Export data

TRANSFORM

Add

Previous steps

> Custom Transform

> Custom formula

> Encode categorical

> Featureize date/time

> Featureize text

> Format string

> Group by

> Handle missing

> Handle outliers

> Manage columns

> Manage rows

> Manage vectors

> Parse column as type

> Process numeric

> Search and edit

Find, replace, split, and otherwise transform input string values using search and edit functions. [Learn more](#)

Transform

Find and replace substring

Replace a substring matching the given regex with a new one.

Input column

race

Pattern

\?

Replacement string

Other

Output column

Optional

Clear

Preview

Add

encounter_id_0 (long)	patient_nbr_0 (long)	race (string)	gender (string)	age (string)	weight (string)	encounter_id_1 (long)	patient_nbr_1 (long)	admission_type_id (la...	discharge_disposition_...	admission_source
2278352	8222157	Caucasian	Female	[0-10]	?	2278352	8222157	6	25	1
149190	55629189	Caucasian	Female	[10-20]	?	149190	55629189	1	1	7
64410	86047875	AfricanAmerican	Female	[20-30]	?	64410	86047875	1	1	7
500364	82442376	Caucasian	Male	[20-40]	?	500364	82442376	1	1	7
16680	42519367	Caucasian	Male	[40-50]	?	16680	42519367	1	1	7
35754	82637451	Caucasian	Male	[50-60]	?	35754	82637451	2	1	2
55842	84259809	Caucasian	Male	[60-70]	?	55842	84259809	3	1	2
63768	114882984	Caucasian	Male	[70-80]	?	63768	114882984	1	1	7
12522	48330783	Caucasian	Female	[80-90]	?	12522	48330783	2	1	4
15738	63555939	Caucasian	Female	[90-100]	?	15738	63555939	3	3	4
28236	89869032	AfricanAmerican	Female	[40-50]	?	28236	89869032	1	1	7
36900	77391171	AfricanAmerican	Male	[60-70]	?	36900	77391171	2	1	4
40936	85504905	Caucasian	Female	[40-50]	?	40936	85504905	1	3	7
43570	77586282	Caucasian	Male	[80-90]	?	43570	77586282	1	6	7
62256	49726791	AfricanAmerican	Female	[60-70]	?	62256	49726791	3	1	2
73578	86328819	AfricanAmerican	Male	[60-70]	?	73578	86328819	1	3	7
77076	92519352	AfricanAmerican	Male	[50-60]	?	77076	92519352	1	1	7
84222	108662661	Caucasian	Female	[50-60]	?	84222	108662661	1	1	7
89682	107389323	AfricanAmerican	Male	[70-80]	?	89682	107389323	3	1	7
148530	69422211	Other	Male	[70-80]	?	148530	69422211	3	6	2
150006	22864151	Other	Female	[50-60]	?	150006	22864151	2	1	4
150048	21239181	Other	Male	[60-70]	?	150048	21239181	2	1	4
182796	63000108	AfricanAmerican	Female	[70-80]	?	182796	63000108	2	1	4
183930	107400762	Caucasian	Female	[80-90]	?	183930	107400762	2	6	1
216156	62718876	AfricanAmerican	Female	[70-80]	?	216156	62718876	3	1	2
221634	21861756	Other	Female	[50-60]	?	221634	21861756	1	1	7
236316	40523101	Caucasian	Male	[80-90]	?	236316	40523101	1	3	7
248916	115196778	Caucasian	Female	[50-60]	?	248916	115196778	1	1	1
250872	41606064	Caucasian	Male	[20-30]	?	250872	41606064	2	1	2
252822	18196454	Caucasian	Female	[80-90]	?	252822	18196454	1	2	7
253380	56480238	AfricanAmerican	Female	[60-70]	?	253380	56480238	1	1	7
253722	96664626	AfricanAmerican	Male	[70-80]	?	253722	96664626	1	5	7
260166	80845353	Caucasian	Female	[70-80]	?	260166	80845353	1	1	7
293058	114715242	Caucasian	Male	[60-70]	?	293058	114715242	2	6	2
293118	3327282	Caucasian	Female	[70-80]	?	293118	3327282	2	11	2

3. Once reviewed, click `Add` to add the transform to your data-flow.

4. Repeat the same technique for other features to replace `weight` , `payer_code` with `0` and `medical_specialty` with `Other` as shown below.

localhost:6419

14/30

Previewing Search and edit

Find and replace substring - Transform: 1st Join

Data

Analysis

Export data

encounter_id_0 (long)	patient_nbr_0 (long)	race (string)	gender (string)	age (string)	weight (string)	encounter_id_1 (long)	patient_nbr_1 (long)	admission_type_id (long)	discharge_disposition_...	admission_source...
2278392	8222157	Caucasian	Female	[0-10)	0	2278392	8222157	6	25	1
149190	55629189	Caucasian	Female	[10-20)	0	149190	55629189	1	1	7
64410	86047875	African/American	Female	[20-30)	0	64410	86047875	1	1	7
500364	82442376	Caucasian	Male	[30-40)	0	500364	82442376	1	1	7
16680	42519267	Caucasian	Male	[40-50)	0	16680	42519267	1	1	7
35754	82637451	Caucasian	Male	[50-60)	0	35754	82637451	2	1	2
55842	84259809	Caucasian	Male	[60-70)	0	55842	84259809	3	1	2
63768	114882984	Caucasian	Male	[70-80)	0	63768	114882984	1	1	7
12522	48330783	Caucasian	Female	[80-90)	0	12522	48330783	2	1	4
15738	63555939	Caucasian	Female	[90-100)	0	15738	63555939	3	3	4
28236	89869032	African/American	Female	[40-50)	0	28236	89869032	1	1	7
36900	77391171	African/American	Male	[60-70)	0	36900	77391171	2	1	4
40926	85504905	Caucasian	Female	[40-50)	0	40926	85504905	1	3	7
42570	77586282	Caucasian	Male	[80-90)	0	42570	77586282	1	6	7
62256	49726791	African/American	Female	[60-70)	0	62256	49726791	3	1	2
73578	86328819	African/American	Male	[60-70)	0	73578	86328819	1	3	7
77076	92519352	African/American	Male	[50-60)	0	77076	92519352	1	1	7
84222	108662661	Caucasian	Female	[50-60)	0	84222	108662661	1	1	7
89682	107389323	African/American	Male	[70-80)	0	89682	107389323	1	1	7
148530	69422211	Other	Male	[70-80)	0	148530	69422211	3	6	2
150006	22864131	Other	Female	[50-60)	0	150006	22864131	2	1	4
150048	21239181	Other	Male	[60-70)	0	150048	21239181	2	1	4
182796	63000108	African/American	Female	[70-80)	0	182796	63000108	2	1	4
183930	107400762	Caucasian	Female	[80-90)	0	183930	107400762	2	6	1
216156	62718876	African/American	Female	[70-80)	0	216156	62718876	3	1	2
221634	21861756	Other	Female	[50-60)	0	221634	21861756	1	1	7
236316	40523301	Caucasian	Male	[80-90)	0	236316	40523301	1	3	7
248916	115196778	Caucasian	Female	[50-60)	0	248916	115196778	1	1	1
250872	41606064	Caucasian	Male	[20-30)	0	250872	41606064	2	1	2
252822	18196434	Caucasian	Female	[80-90)	0	252822	18196434	1	2	7
253380	56480238	African/American	Female	[60-70)	0	253380	56480238	1	1	7
253722	96664626	African/American	Male	[70-80)	0	253722	96664626	1	5	7
260166	80845353	Caucasian	Female	[70-80)	0	260166	80845353	1	1	7
293058	114715242	Caucasian	Male	[60-70)	0	293058	114715242	2	6	2
293118	3327282	Caucasian	Female	[70-80)	0	293118	3327282	2	11	2

TRANSFORM

Add

Previous steps

> Custom Transform

> Custom formula

> Encode categorical

> Featureize date/time

> Featureize text

> Format string

> Group by

> Handle missing

> Handle outliers

> Manage columns

> Manage rows

> Manage vectors

> Parse column as type

> Process numeric

> Search and edit

Find, replace, split, and otherwise transform input string values using search and edit functions. [Learn more](#)

Transform

Find and replace substring

Replace a substring matching the given regex with a new one.

Input column

weight

Pattern

17

Replacement string

0

Output column

Optional

Clear

Preview

Add

Previewing Search and edit

Find and replace substring - Transform: 1st Join

Data

Analysis

Export data

	weight (string)	encounter_id_1 (long)	patient_nbr_1 (long)	admission_type_id (long)	discharge_disposition_...	admission_source_id (long)	time_in_hospital (long)	payer_code (string)	medical_specialty (string)	num_lab_procedures (long)
0	2278392	8222157	6	25	1	1	1	0	Pediatrics-Endocrinology	41
0	149190	55629189	1	1	7	3	0	7	7	59
0	64410	86047875	1	1	7	2	0	7	7	11
0	500364	82442376	1	1	7	2	0	7	7	44
0	16680	42519267	1	1	7	1	0	7	7	51
0	35754	82637451	2	1	2	3	0	7	7	31
0	55842	84259809	3	1	2	4	0	7	7	70
0	63768	114882984	1	1	7	5	0	7	7	78
0	12522	48330783	2	1	4	13	0	7	7	68
0	15738	63555939	3	3	4	12	0	InternalMedicine	33	47
0	28236	89869032	1	1	7	9	0	7	7	37
0	36900	77391171	2	1	4	7	0	7	7	62
0	40926	85504905	1	3	7	7	0	Family/GeneralPractice	60	49
0	42570	77586282	1	6	7	10	0	Family/GeneralPractice	55	75
0	62256	49726791	3	1	2	1	0	7	7	45
0	73578	86328819	1	3	7	12	0	7	7	45
0	77076	92519352	1	1	7	4	0	7	7	29
0	84222	108662661	1	1	7	3	0	Cardiology	35	42
0	89682	107389323	1	1	7	5	0	7	7	66
0	148530	69422211	3	6	2	6	0	7	7	36
0	150006	22864131	2	1	4	2	0	7	7	47
0	150048	21239181	2	1	4	2	0	7	7	42
0	182796	63000108	2	1	4	2	0	7	7	19
0	183930	107400762	2	6	1	11	0	7	7	33
0	216156	62718876	3	1	2	3	0	7	7	64
0	221634	21861756	1	1	7	1	0	7	7	25
0	236316	40523301	1	3	7	6	0	Cardiology	53	87
0	248916	115196778	1	1	1	2	0	Surgery-General	52	53
0	250872	41606064	2	1	2	10	0	7	7	37
0	252822	18196434	1	2	7	5	0	Cardiology	27	37
0	253380	56480238	1	1	7	6	0	7	7	46
0	253722	96664626	1	5	7	1	0	7	7	46
0	260166	80845353	1	1	7	6	0	Family/GeneralPractice	37	46
0	293058	114715242	2	6	2	5	0	7	7	46
0	293118	3327282	2	11	2	5	0	7	7	46

TRANSFORM

Add

Previous steps

> Custom Transform

> Custom formula

> Encode categorical

> Featureize date/time

> Featureize text

> Format string

> Group by

> Handle missing

> Handle outliers

> Manage columns

> Manage rows

> Manage vectors

> Parse column as type

> Process numeric

> Search and edit

Find, replace, split, and otherwise transform input string values using search and edit functions. [Learn more](#)

Transform

Find and replace substring

Replace a substring matching the given regex with a new one.

Input column

payer_code

Pattern

17

Replacement string

0

Output column

Optional

Clear

Preview

Add

Previewing Search and edit

Find and replace substring · Transform: 1st Join

Data

Analysis

Export data

	weight (string)	encounter_id_1 (long)	patient_nbr_1 (long)	admission_type_id (long)	discharge_disposition_1 (long)	admission_source_id (long)	time_in_hospital (long)	payer_code (string)	medical_specialty (string)	num_lab_procedures (long)
0	2278392	8222157	6	25	1	1	0	Pediatrics-Endocrinology	41	
0	149190	55629189	1	1	7	3	0	Other	59	
0	64410	86047875	1	1	7	2	0	Other	11	
0	500364	82442376	1	1	7	2	0	Other	44	
0	16680	42519267	1	1	7	1	0	Other	51	
0	35754	82637451	2	1	2	3	0	Other	31	
0	55842	84259809	3	1	2	4	0	Other	70	
0	63768	114882984	1	1	7	5	0	Other	73	
0	12522	48330783	2	1	4	13	0	Other	68	
0	15738	63555939	3	3	4	12	0	InternalMedicine	33	
0	28236	89869032	1	1	7	9	0	Other	47	
0	36900	77391171	2	1	4	7	0	Other	62	
0	40926	85504905	1	3	7	7	0	Family/GeneralPractice	60	
0	42570	77586282	1	6	7	10	0	Family/GeneralPractice	55	
0	62256	49726791	3	1	2	1	0	Other	49	
0	73578	86328819	1	3	7	12	0	Other	75	
0	77076	92519352	1	1	7	4	0	Other	45	
0	84222	108662661	1	1	7	3	0	Cardiology	29	
0	89682	107389323	1	1	7	5	0	Other	35	
0	148530	69422211	3	6	2	6	0	Other	42	
0	150006	22864131	2	1	4	2	0	Other	66	
0	150048	21239181	2	1	4	2	0	Other	36	
0	182796	63000108	2	1	4	2	0	Other	47	
0	183930	107400762	2	6	1	11	0	Other	42	
0	216156	62718876	3	1	2	3	0	Other	19	
0	221634	21861756	1	1	7	1	0	Other	33	
0	236316	40523301	1	3	7	6	0	Cardiology	64	
0	248916	115196778	1	1	1	2	0	Surgery-General	25	
0	250872	41606064	2	1	2	10	0	Other	53	
0	252822	18196434	1	2	7	5	0	Cardiology	52	
0	253380	56480238	1	1	7	6	0	Other	87	
0	253722	96664626	1	5	7	1	0	Other	53	
0	260166	80845353	1	1	7	6	0	Family/GeneralPractice	27	
0	293058	114715242	2	6	2	5	0	Other	37	
0	293118	3327282	2	11	2	5	0	Other	46	

TRANSFORM

Add

Previous steps

> Custom Transform

> Custom formula

> Encode categorical

> Featureize date/time

> Featureize text

> Format string

> Group by

> Handle missing

> Handle outliers

> Manage columns

> Manage rows

> Manage vectors

> Parse column as type

> Process numeric

> Search and edit

Find, replace, split, and otherwise transform input string values using search and edit functions. [Learn more.](#)

Transform

Find and replace substring

Replace a substring matching the given regex with a new one.

Input column

medical_specialty

Pattern

/?

Replacement string

Other

Output column

Optional

Clear

Preview

Add

One-hot Encoding for categorical features

1. Pick `Encode categorical` from the list of transforms on the right panel. Select `One-hot encode` and `race` for input column. For `Output style`, choose `Columns`. After filling the fields click `Preview`
2. After review the transformation results, Click `Add` to add the change to the data flow

Previewing Encode categorical

Find and replace substring · Transform: 1st Join

Data

Analysis

Export data

_2 (long)	diag_3 (long)	readmitted (string)	race_Caucasian (float)	race_AfricanAmerican ...	race_Other (float)	race_Hispanic (float)	race_Asian (float)
		NO	1	0	0	0	0
	255	>30	1	0	0	0	0
		NO	0	1	0	0	0
	403	NO	1	0	0	0	0
	250	NO	1	0	0	0	0
	250	>30	1	0	0	0	0
		NO	1	0	0	0	0
	250	>30	1	0	0	0	0
	38	NO	1	0	0	0	0
	486	NO	1	0	0	0	0
	996	>30	0	1	0	0	0
	197	<30	0	1	0	0	0
	250	<30	1	0	0	0	0
	427	NO	1	0	0	0	0
	627	>30	0	1	0	0	0
	996	NO	0	1	0	0	0
	414	<30	0	1	0	0	0
	250	NO	1	0	0	0	0
	416	>30	0	1	0	0	0
	714	NO	0	0	1	0	0
	428	NO	0	0	1	0	0
	427	NO	0	0	1	0	0
	582	NO	0	1	0	0	0

TRANSFORM

Add

Previous steps

> Custom Transform

> Custom formula

> Encode categorical

Convert categorical variables to numeric or vector representations. [Learn more.](#)

Transform

One-hot encode

Input column

race

☐ Input already ordinal encoded

Invalid handling strategy

Keep

☐ Drop last

Output style

Columns

Output column

Optional

Clear

Preview

Add

3. Repeat above 2 steps for `age` and `medical_specialty_filler` to one-hot encode those categorical features as well.

Ordinal Encoding for categorical features

1. Pick `Encode categorical` from the list of transforms on the right panel. Select `Ordinal encode` and `gender` for input column. For `Invalid handling strategy` select `skip`. After filling the fields click `Preview`

Previewing Encode categorical

One-hot encode · Transform: 1st Join

Data

Analysis

Export data

encounter_id_0 (long)	patient_nbr_0 (long)	gender (float)	weight (string)	encounter_id_1 (long)	patient_nbr_1 (long)	admission_type_id (lo...	discharge_disposition
2278392	8222157	0	0	2278392	8222157	6	25
149190	55629189	0	0	149190	55629189	1	1
64410	86047875	0	0	64410	86047875	1	1
500364	82442376	1	0	500364	82442376	1	1
16680	42519267	1	0	16680	42519267	1	1
35754	82637451	1	0	35754	82637451	2	1
55842	84259809	1	0	55842	84259809	3	1
63768	114882984	1	0	63768	114882984	1	1
12522	48330783	0	0	12522	48330783	2	1
15738	63555939	0	0	15738	63555939	3	3
28236	89869032	0	0	28236	89869032	1	1
36900	77391171	1	0	36900	77391171	2	1
40926	85504905	0	0	40926	85504905	1	3
42570	77586282	1	0	42570	77586282	1	6
62256	49726791	0	0	62256	49726791	3	1
73578	86328819	1	0	73578	86328819	1	3
77076	92519352	1	0	77076	92519352	1	1
84222	108662661	0	0	84222	108662661	1	1

TRANSFORM

Add

Previous steps

> Custom Transform

> Custom formula

> Encode categorical

Convert categorical variables to numeric or vector representations. [Learn more](#)

Transform

Ordinal encode

Input column

gender

Output column

Optional

Invalid handling strategy

Keep

Clear

Preview

Add

Custom Transformations – Add new features to your dataset

If we choose to store our transformed features into Amazon SageMaker Feature Store, a pre-requisite is to insert Event-Time feature into the dataset. We can easily do that using Custom Transformations

1. Pick **Custom Transform** from the list of transforms on the right panel
2. select **Python (Pandas)** and enter below line of code in the text box. Then click **Preview** to view the results.

```
# Table is available as variable `df`
import time
df['eventTime'] = time.time()
```

Back to data flow

Previewing Python (Pandas)

Ordinal encode · Transform: 1st Join

Data

Analysis

Export data

encounter_id_0 (long)	patient_nbr_0 (long)	gender (float)	weight (string)	encounter_id_1 (long)	patient_nbr_1 (long)	admission_type_id (lo...	discharge_disposition	gender_ordinal (float)	eventTime (float)
2278392	8222157	0	0	2278392	8222157	6	25	0	1615475511.237896
149190	55629189	0	0	149190	55629189	1	1	0	1613473311.237896
64410	86047875	0	0	64410	86047875	1	1	0	1613473311.237896
500364	82442376	1	0	500364	82442376	1	1	0	1613473311.237896
16680	42519267	1	0	16680	42519267	1	1	0	1613473311.237896
35754	82637451	1	0	35754	82637451	2	1	0	1613473311.237896
55842	84259809	1	0	55842	84259809	3	1	0	1613473311.237896
63768	114882984	1	0	63768	114882984	1	1	0	1613473311.237896
12522	48330783	0	0	12522	48330783	2	1	0	1613473311.237896
15738	63555939	0	0	15738	63555939	3	3	0	1613473311.237896
28236	89869032	0	0	28236	89869032	1	1	0	1613473311.237896
36900	77391171	1	0	36900	77391171	2	1	0	1613473311.237896
40926	85504905	0	0	40926	85504905	1	3	0	1613473311.237896
42570	77586282	1	0	42570	77586282	1	6	0	1613473311.237896
62256	49726791	0	0	62256	49726791	3	1	0	1613473311.237896
73578	86328819	1	0	73578	86328819	1	3	0	1613473311.237896
77076	92519352	1	0	77076	92519352	1	1	0	1613473311.237896
84222	108662661	0	0	84222	108662661	1	1	0	1613473311.237896

TRANSFORM

Add

Previous steps

> Custom Transform

Using Python (Pandas) requires your dataset to fit in memory and only uses a single instance in batch computation. It is ideal for smaller datasets less than 2GB and experimentation but we recommend Python (PySpark) for production use cases.

Python (Pandas)

1 # Table is available as variable `df`
2 import time
3 df['eventTime'] = time.time()

Clear

Preview

Add

> Custom formula

3. Click **Add** to add the change to the data flow

Transform the target Label

The target label readmitted has 3 classes: NO readmission, readmitted <30 days and readmitted >30 days. We saw in our **Histogram** analysis that there is a strong class imbalance as majority of the patients did not readmit. We could combine the latter two classes into a positive class to denote the patients being readmitted, and turn the classification problem into a binary case instead of multi-class. Let's use **Search and Edit Transform** to convert string values to binary values.

1. Pick `Search and edit` from the list of transforms on the right panel. Select `Find and replace substring`
2. Select the target column `readmitted` for `Input column` and use `>30|<30` regex for `Pattern` . For the `Replacement String` use `1` .
3. So, here we are converting all the values that have either `>30` or `<30` values to `1` . After making your config selections, hit `Preview` to review the converted column as shown below.

Previewing Search and edit

Ordinal encode - Transform: 1st Join

Data

Analysis

Export data

diag_3 (long)	readmitted (string)	race_Caucasian (float)	race_AfricanAmerican ...	race_Other (float)	race_Hispanic (float)	race_Asian (float)	age_70-80 (float)	age_50-70 (float)	age_50-60 (float)	age_90-90 (fl...
255	NO	1	0	0	0	0	0	0	0	0
	1	1	0	0	0	0	0	0	0	0
403	NO	1	0	0	0	0	0	0	0	0
250	NO	1	0	0	0	0	0	0	0	0
250	1	1	0	0	0	0	0	0	1	0
	NO	1	0	0	0	0	0	1	0	0
250	1	1	0	0	0	0	1	0	0	0
38	NO	1	0	0	0	0	0	0	0	1
486	NO	1	0	0	0	0	0	0	0	0
996	1	0	1	0	0	0	0	0	0	0
197	1	0	1	0	0	0	0	1	0	0
250	1	1	0	0	0	0	0	0	0	0
427	NO	1	0	0	0	0	0	0	0	1
627	1	0	1	0	0	0	0	1	0	0
996	NO	0	1	0	0	0	0	1	0	0
414	1	0	1	0	0	0	0	0	1	0
250	NO	1	0	0	0	0	0	0	1	0
416	1	0	1	0	0	0	1	0	0	0
714	NO	0	0	1	0	0	1	0	0	0
428	NO	0	0	1	0	0	0	0	1	0
427	NO	0	0	1	0	0	0	1	0	0
582	NO	0	1	0	0	0	1	0	0	0
	1	1	0	0	0	0	0	0	0	1
427	NO	0	1	0	0	0	1	0	0	0
250	NO	0	0	1	0	0	0	0	1	0
414	NO	1	0	0	0	0	0	0	0	1
250	1	1	0	0	0	0	0	0	1	0
263	1	1	0	0	0	0	0	0	0	0
414	1	1	0	0	0	0	0	0	0	1
250	NO	0	1	0	0	0	0	1	0	0
276	1	0	1	0	0	0	1	0	0	0
250	1	1	0	0	0	0	1	0	0	0
482	1	1	0	0	0	0	0	1	0	0
414	NO	1	0	0	0	0	1	0	0	0

TRANSFORM

Add

Previous steps (7)

> Custom Transform

> Custom formula

> Encode categorical

> Featureize date/time

> Featureize text

> Format string

> Group by

> Handle missing

> Handle outliers

> Manage columns

> Manage rows

> Manage vectors

> Parse column as type

> Process numeric

> Search and edit

Find, replace, split, and otherwise transform input string values using search and edit functions. [Learn more](#)

Transform

Find and replace substring

Replace a substring matching the given regex with a new one.

Input column

readmitted

Pattern

>30<30

Replacement string

1

Output column

Optional

Clear

Preview

Add

4. Once reviewed, click `Add` to add the transform to your data-flow.
5. Let's repeat the same to convert `NO` values to `0` . Pick `Search and edit` from the list of transforms on the right panel.
6. Choose `Find and replace substring` transform. Select the target column `readmitted` for `Input column` and use `NO` regex for `Pattern` . For the `Replacement String` use `0` .
7. After making your config selections, hit `Preview` to review the converted column as shown below.

Previewing Search and edit

Find and replace substring · Transform: 1st Join

Data Analysis

Export data

diag_3 (long)	readmitted (string)	race_Caucasian (float)	race_AfricanAmerican ...	race_Other (float)	race_Hispanic (float)	race_Asian (float)	age_70-80 (float)	age_50-70 (float)	age_50-60 (float)	age_80-90 (float)
255	1	1	0	0	0	0	0	0	0	0
403	0	1	0	0	0	0	0	0	0	0
250	0	1	0	0	0	0	0	0	0	0
250	1	1	0	0	0	0	0	0	1	0
250	0	1	0	0	0	0	0	1	0	0
250	1	1	0	0	0	0	1	0	0	0
38	0	1	0	0	0	0	0	0	0	1
486	0	1	0	0	0	0	0	0	0	0
996	1	0	1	0	0	0	0	0	0	0
197	0	0	1	0	0	0	0	1	0	0
250	1	1	0	0	0	0	0	0	0	0
427	0	1	0	0	0	0	0	0	0	1
627	1	0	1	0	0	0	0	1	0	0
996	0	0	1	0	0	0	0	1	0	0
414	1	0	1	0	0	0	0	0	1	0
250	0	1	0	0	0	0	0	0	1	0
416	1	0	1	0	0	0	1	0	0	0
714	0	0	0	1	0	0	1	0	0	0
428	0	0	0	1	0	0	0	0	1	0
427	0	0	0	1	0	0	0	1	0	0
582	0	0	1	0	0	0	1	0	0	0
427	1	1	0	0	0	0	0	0	0	1
250	0	0	0	1	0	0	1	0	0	0
414	0	1	0	0	0	0	0	0	1	0
250	1	1	0	0	0	0	0	0	1	0
263	1	1	0	0	0	0	0	0	0	0
414	1	1	0	0	0	0	0	0	0	1
250	0	0	1	0	0	0	0	1	0	0
276	1	0	1	0	0	0	1	0	0	0
250	1	1	0	0	0	0	1	0	0	0
482	1	1	0	0	0	0	0	1	0	0
414	0	1	0	0	0	0	1	0	0	0

TRANSFORM

Add Previous steps (7)

- Custom Transform
- Custom formula
- Encode categorical
- Featurize date/time
- Featurize text
- Format string
- Group by
- Handle missing
- Handle outliers
- Manage columns
- Manage rows
- Manage vectors
- Parse column as type
- Process numeric
- Search and edit

Find, replace, split, and otherwise transform input string values using search and edit functions. [Learn more.](#)

Transform

Find and replace substring

Replace a substring matching the given regex with a new one.

Input column

readmitted

Pattern

NO

Replacement string

0

Output column

Optional

Clear Preview Add

5. Once reviewed, click **Add** to add the transform to your data-flow. Now our target label is ready for ML.

Position the target label as first column to utilize XGBoost algorithm

As we are going to use XGBoost built-in SageMaker algorithm to train the model, the algorithm assumes that the target label is in the first column. Let's do that.

1. Pick **Manage Column** from the list of transforms on the right panel. Select **Move Column** for Transform and select **Move** to start for Move type. Provide a name to new column **readmitted**. After filling the fields click **Preview**
2. After reviewing the transformation results, Click **Add** to add the change to your data flow

Previewing Manage columns

Find and replace substring · Transform: 1st Join

Data Analysis

Export data

readmitted (string)	encounter_id_0 (long)	patient_nbr_0 (long)	gender (float)	weight (string)	encounter_id_1 (long)	patient_nbr_1 (long)	admission_type_id (lo...	dischar
0	2278392	8222157	0	0	2278392	8222157	6	25
1	149190	55629189	0	0	149190	55629189	1	1
0	64410	86047875	0	0	64410	86047875	1	1
0	500364	82442376	1	0	500364	82442376	1	1
0	16680	42519267	1	0	16680	42519267	1	1
1	35754	82637451	1	0	35754	82637451	2	1
0	55842	84259809	1	0	55842	84259809	3	1
1	63768	114882984	1	0	63768	114882984	1	1
0	12522	48330783	0	0	12522	48330783	2	1
0	15738	63555939	0	0	15738	63555939	3	3
1	28236	89869032	0	0	28236	89869032	1	1
1	36900	77391171	1	0	36900	77391171	2	1
1	40926	85504905	0	0	40926	85504905	1	3
0	42570	77586282	1	0	42570	77586282	1	6
1	62256	49726791	0	0	62256	49726791	3	1
0	73578	86328819	1	0	73578	86328819	1	3
1	77076	92519352	1	0	77076	92519352	1	1
0	84222	108662661	0	0	84222	108662661	1	1
1	89682	107389323	1	0	89682	107389323	1	1
0	148530	69422211	1	0	148530	69422211	3	6
0	150006	22864131	0	0	150006	22864131	2	1
0	150048	21239181	1	0	150048	21239181	2	1
0	182796	63000108	0	0	182796	63000108	2	1
1	183930	107400762	0	0	183930	107400762	2	6

TRANSFORM

Add Previous steps (7)

- Custom Transform
- Custom formula
- Encode categorical
- Featurize date/time
- Featurize text
- Format string
- Group by
- Handle missing
- Handle outliers
- Manage columns

Move, drop, duplicate or rename columns in the dataset. [Learn more.](#)

Transform

Move column

Move type

Move to start

Moves the column so it is the first column in the dataset.

Column to move

readmitted

Clear Preview Add

Let's Drop redundant columns

1. Pick `Manage Columns` from the list of transforms on the right panel
2. Choose `Drop Column` transform and select `encounter_id_0` for column to drop

Previewing Manage columns

Move column - Transform: 1st Join

Data

Analysis

Export data

readmitted (string)	patient_nbr_0 (long)	gender (float)	weight (string)	encounter_id_1 (long)	patient_nbr_1 (long)	admission_type_id (lo...	discharge_disposition
0	8222157	0	0	2278392	8222157	6	25
1	55629189	0	0	149190	55629189	1	1
0	86047875	0	0	64410	86047875	1	1
0	82442376	1	0	500364	82442376	1	1
0	42519267	1	0	16680	42519267	1	1
1	82637451	1	0	35754	82637451	2	1
0	84259809	1	0	55842	84259809	3	1
1	114882984	1	0	63768	114882984	1	1
0	48330783	0	0	12522	48330783	2	1
0	63555939	0	0	15738	63555939	3	3
1	89869032	0	0	28236	89869032	1	1
1	77391171	1	0	36900	77391171	2	1
1	85504905	0	0	40926	85504905	1	3
0	77586282	1	0	42570	77586282	1	6
1	49726791	0	0	62256	49726791	3	1
0	86328819	1	0	73578	86328819	1	3
1	92519352	1	0	77076	92519352	1	1
0	108662661	0	0	84222	108662661	1	1
1	107389323	1	0	89682	107389323	1	1
0	69422211	1	0	148530	69422211	3	6
0	22864131	0	0	150006	22864131	2	1

TRANSFORM

Add

Previous steps (10)

> Custom Transform

> Custom formula

> Encode categorical

> Featurize date/time

> Featurize text

> Format string

> Group by

> Handle missing

> Handle outliers

> Manage columns

Move, drop, duplicate or rename columns in the dataset. [Learn more.](#)

Transform

Drop column

Column to drop

encounter_id_0

Clear

Preview

Add

3. Click `Preview` to preview the changes to the data set. Then `Add` to add the changes to flow file.
4. Repeat above steps 1 through 3 for other redundant columns `patient_nbr_0` , `encounter_id_1` , `patient_nbr_1` .

At this stage, we have done a few analyses and applied a few transformations on our raw input dataset. If we choose to preserve the transformed state of the input dataset, kind a like checkpoint, you can do that using the `Export data` button shown below. This option will allow you to persist the transformed dataset on to an Amazon S3 bucket.

< Back to data flow

Drop column - Transform: 1st Join

Data

Analysis

Export data

readmitted_converted...	weight (string)	admission_type_id (lo...	discharge_disposition_...	admission_source_id (l...	time_in_hospital (long)	payer_code (string)
0	0	6	25	1	1	0
1	0	1	1	7	3	0
0	0	1	1	7	2	0
0	0	1	1	7	2	0
0	0	1	1	7	1	0
1	0	2	1	2	3	0

TRANSFORM

Add

Previous steps

> Custom Transform

> Custom formula

> Encode categorical

> Featurize date/time

> Featurize text

dw-workshop.flow

< Back to transform

Export data

Export data

Export your data in your current session to S3.

S3 location

s3://agemaker-us-east-1-xxxxxx/

Browse

File type

CSV (*.csv)

Delimiter

Comma (,)

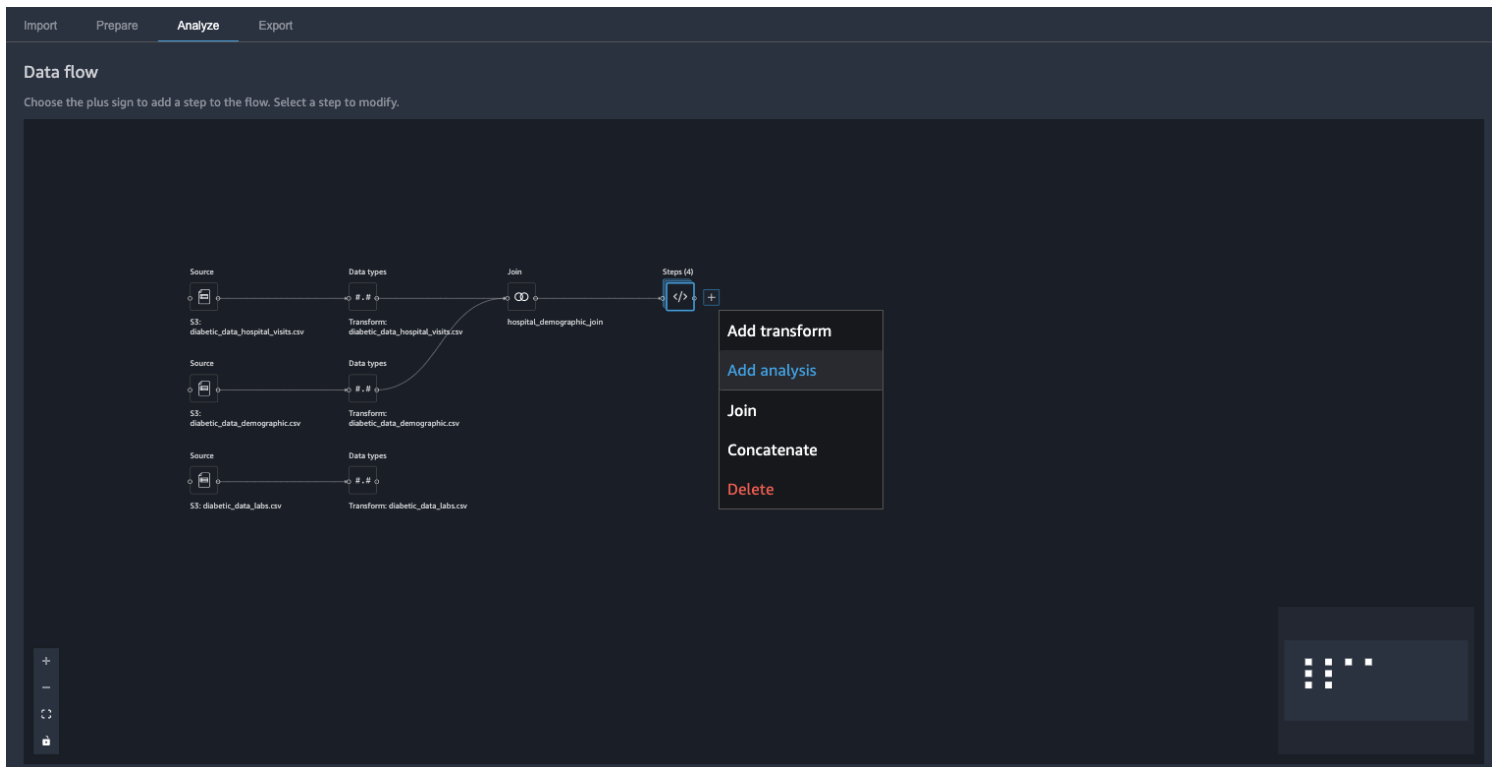
Compression

None

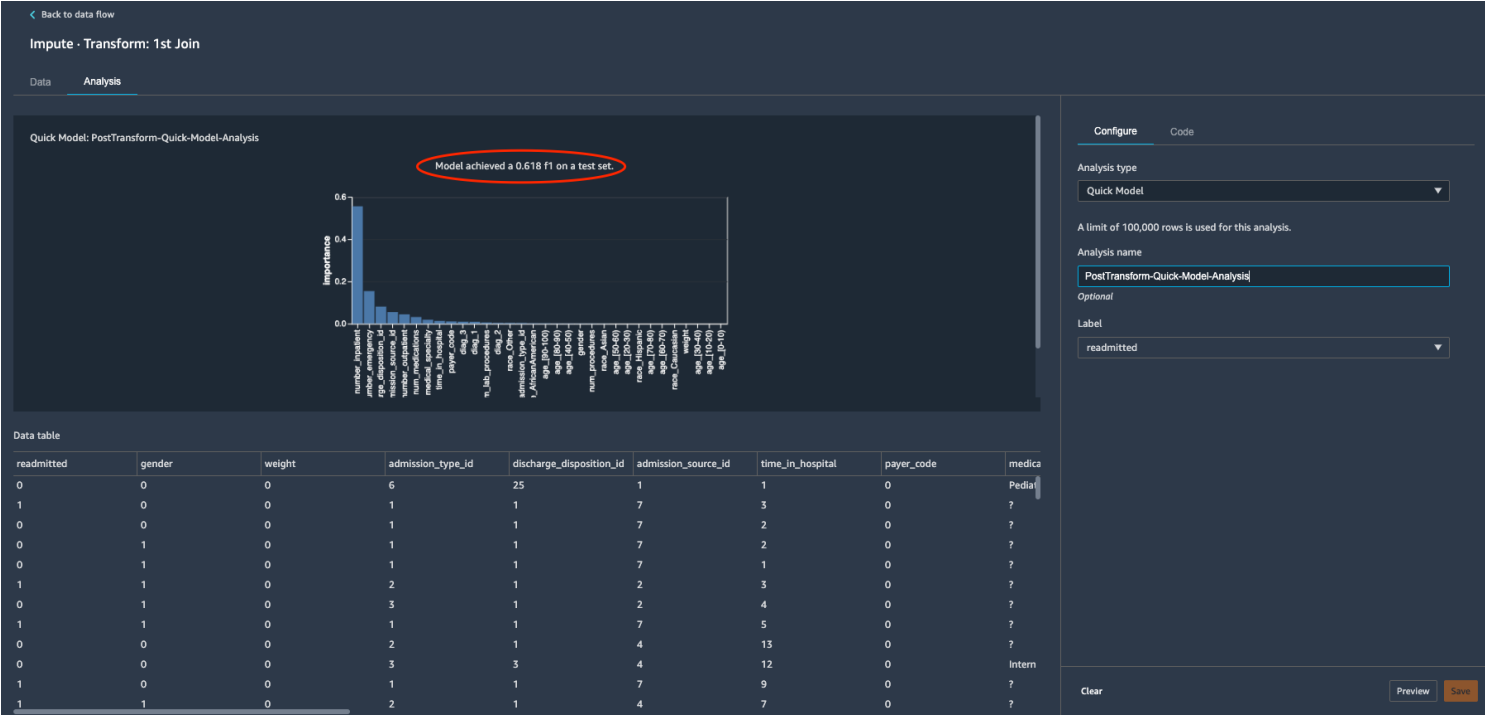
Quick Model Analysis

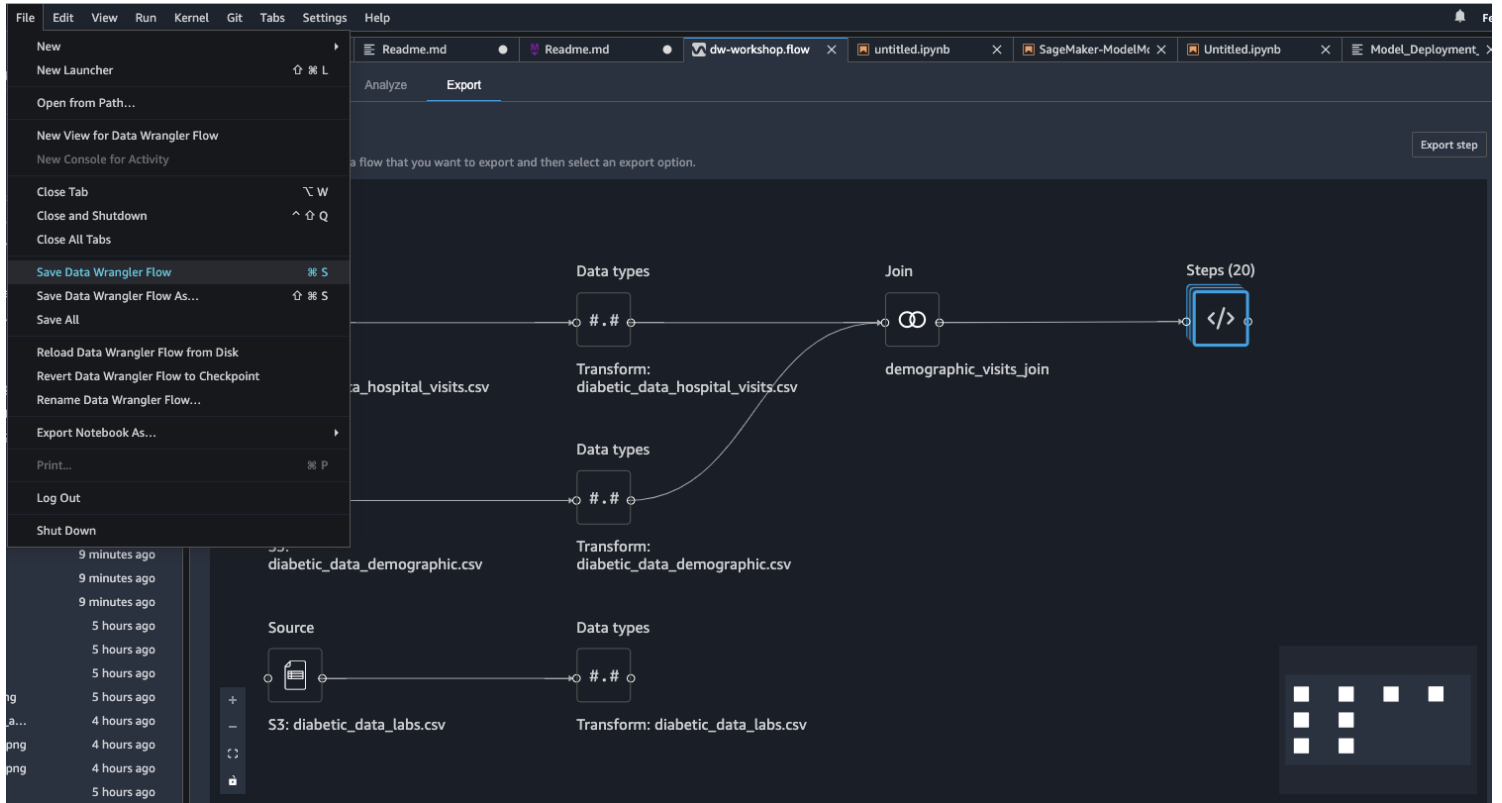
Now that we have applied transformations to our initial dataset, let's explore Quick Model analysis. Quick Model helps to quickly evaluate the training dataset and produce importance scores for each feature. A feature importance score indicates how useful a feature is at predicting a target label. The feature importance score is between [0, 1] and a higher number indicates that the feature is more important to the whole dataset. Since our use-case relates to classification problem type, Quick Model also generates F1 score for current dataset.

1. Click + sign next to Join flow icon and choose **Add analysis**

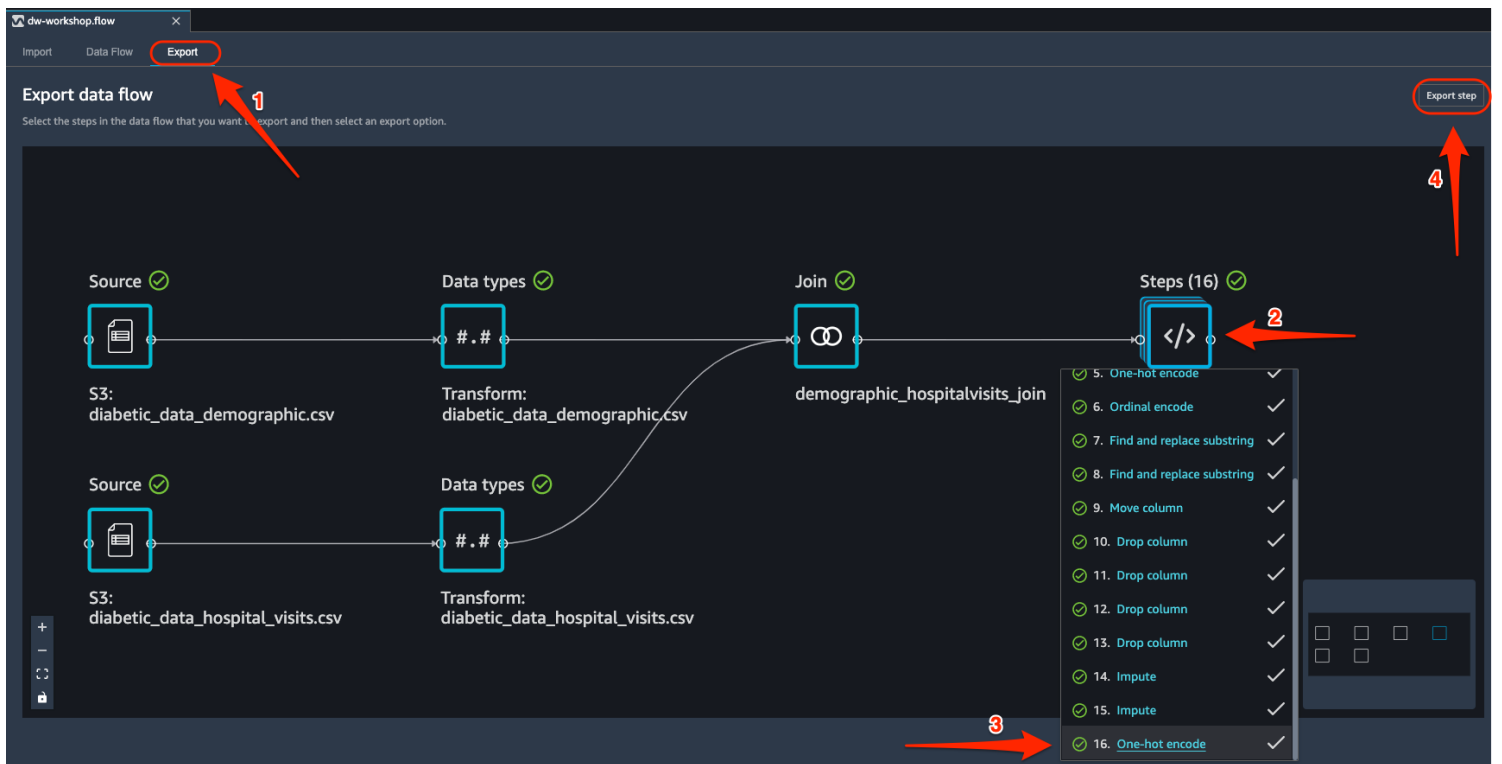


2. Select **Quick Model** from the list of Analysis types on the right panel.
3. Give a name to your analysis and select the target label in **Label** field.
4. Click **Preview** and wait for the model to be results to be displayed on the screen





2. Click **Export** tab and select **Steps** icon to reveal all the DW flow steps. Click the last step to mark it as check (as shown in figure below)



3. Click **Export** step to reveal the export options. You currently have 4 export options

- **Save to S3** Save the data to an S3 bucket using a Amazon SageMaker Processing Job.
- **Pipeline** exports a Jupyter Notebook that creates an Amazon SageMaker Pipeline with your data flow.
- **Python Code** exports your data flow to python code.

- Feature Store exports a Jupyter Notebook that creates an Amazon SageMaker Feature Store feature group and adds features to an offline or online feature store.

You can find more information for each export option in this page.

Export data flow

Select the steps in the data flow that you want to export and then select an export option.

Source ✓
S3: diabetic_data_demographic.csv

Data types ✓
Transform: diabetic_data_demographic.csv

Join ✓
demographic_hospitalvisits_join

Source ✓
S3: diabetic_data_hospital_visits.csv

Data types ✓
Transform: diabetic_data_hospital_visits.csv

Export options:

- Save to S3** (highlighted): Save to S3 using a SageMaker Processing Job.
- Pipeline**: Export a Jupyter Notebook that creates a Pipeline with your data flow.
- Python Code**: Export your data flow to python code.
- Feature Store**: Export a Jupyter Notebook that creates a Feature Store feature group and adds features to an offline or online feature store.

Export step (highlighted)

- Select **Save to S3** to generate a fully implemented Jupyter Notebook that creates a processing Job using your data flow file.

Save to S3 with a SageMaker Processing Job

Quick Start To save your processed data to S3, select the Run menu above and click Run all cells. [View the status of the export job and the output S3 location.](#)

This notebook executes your Data Wrangler Flow `dw-workshop.flow` on the entire dataset using a SageMaker Processing Job and will save the processed data to S3.

This notebook saves data from the step `Manage Columns`. To save from a different step, go to Data Wrangler to select a new step to export.

Contents

- Inputs and Outputs
- Run Processing Job
 - Job Configurations
 - Create Processing Job
 - Job Status & S3 Output Location
- Optional Next Steps
 - Load Processed Data into Pandas
 - Train a model with SageMaker

Inputs and Outputs

The below settings configure the inputs and outputs for the flow export.

Configurable Settings

In **Input - Source** you can configure the data sources that will be used as input by Data Wrangler

- For S3 sources, configure the source attribute that points to the input S3 prefixes
- For all other sources, configure attributes like `query_string`, `database` in the source's `DatasetDefinition` object.

If you modify the inputs the provided data must have the same schema and format as the data used in the Flow. You should also re-execute the cells in this section if you have modified the settings in any data sources.

```
[ ]: from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker.dataset_definition.inputs import AthenaDatasetDefinition, DatasetDefinition, RedshiftDatasetDefinition

data_sources = []
```

3. Processing and Training Jobs for Model building

Processing Job submission

1. We are now ready to submit a SageMaker Processing Job using the data flow file. Run all the cells upto **Create Processing Job** . This cell **Create Processing Job** will trigger a new SageMaker processing job by provisioning managed infrastructure and running the required DataWrangler docker container on that infrastructure.

Create Processing Job

To launch a Processing Job, you will use the SageMaker Python SDK to create a Processor function.

```
[ ]: from sagemaker.processing import Processor
from sagemaker.network import NetworkConfig

processor = Processor(
    role=iam_role,
    image_uri=container_uri,
    instance_count=instance_count,
    instance_type=instance_type,
    volume_size_in_gb=volume_size_in_gb,
    network_config=NetworkConfig(enable_network_isolation=enable_network_isolation),
    sagemaker_session=sess
)

# Start Job
processor.run(
    inputs=[flow_input] + data_sources,
    outputs=[processing_job_output],
    arguments=[f"--output-config '{json.dumps(output_config)}'",
    wait=False,
    logs=False,
    job_name=processing_job_name
)
```

2. You can check the status of the submitted processing job by running next cell **Job Status & S3 Output Location**

Job Status & S3 Output Location

Below you wait for processing job to finish. If it finishes successfully, the raw parameters used by the Processing Job will be printed

```
[ ]: s3_job_results_path = f"s3://{bucket}/{s3_output_prefix}/{processing_job_name}"
print(f"Job results are saved to S3 path: {s3_job_results_path}")

job_result = sess.wait_for_processing_job(processing_job_name)
job_result
```

3. You can also check the status of the submitted processing job from Amazon SageMaker Console as shown below

The screenshot shows the Amazon SageMaker console interface. On the left is a navigation sidebar with 'Processing' selected. The main panel displays a table of processing jobs. The first job is highlighted with a red box.

Name	ARN	Creation time	Duration	Status
data-wrangler-flow-processing-27-20-53-27-4b246dc5	arn:aws:sagemaker:us-east-1:123456789012:processing-job/data-wrangler-flow-processing-27-20-53-27-4b246dc5	Aug 27, 2021 20:54 UTC	2 minutes	InProgress
huggingface-sm-2021-08-25--ProfilerReport-c8f5319b	arn:aws:sagemaker:us-east-1:123456789012:processing-job/huggingface-sm-2021-08-25--ProfilerReport-c8f5319b	Aug 25, 2021 22:47 UTC	12 minutes	Completed
huggingface-sm-2021-08-25--LowGPUUtilization-acdc68ba	arn:aws:sagemaker:us-east-1:123456789012:processing-job/huggingface-sm-2021-08-25--LowGPUUtilization-acdc68ba	Aug 25, 2021 22:47 UTC	12 minutes	Completed

Train a model with Amazon SageMaker

1. Now that the data has been processed, you may want to train a model using the data. The same notebook has sample steps to train a model using Amazon SageMaker built-in XGBoost algorithm. Since our use case is binary classification, we need to change the objective to "binary:logistic" inside the sample training steps as shown below.

Configure the algorithm and training job

The Training Job hyperparameters are set. For more information on XGBoost Hyperparameters, see <https://xgboost.readthedocs.io/en/latest/parameter.html>.

```
[ ]: region = boto3.Session().region_name
container = sagemaker.image_uris.retrieve("xgboost", region, "1.2-1")
hyperparameters = {
    "max_depth": "5",
    "objective": "binary:logistic",
    "num_round": "10",
}
train_content_type = (
    "application/x-parquet" if output_content_type.upper() == "PARQUET"
    else "text/csv"
)
train_input = sagemaker.inputs.TrainingInput(
    s3_data=s3_training_input_path,
    content_type=train_content_type,
)
```

2. All set. Now we are ready to fire our training job using SageMaker managed infrastructure. Run the cell below.

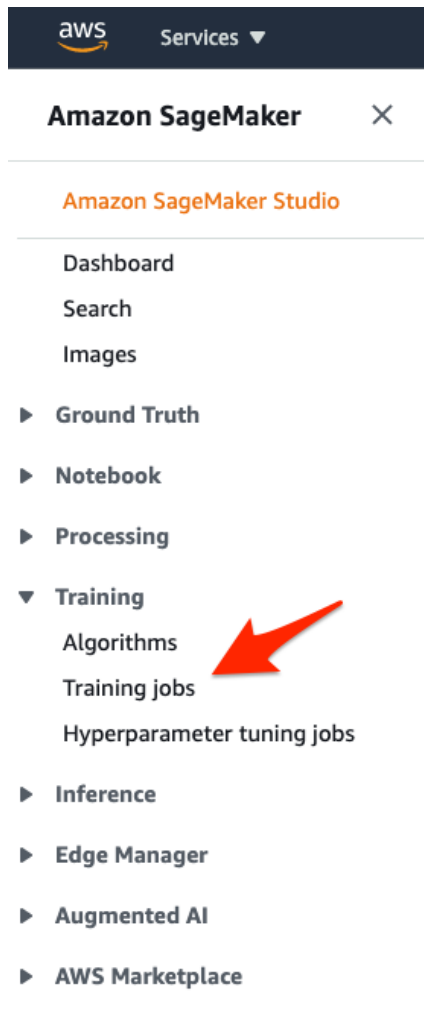
Start the Training Job

The TrainingJob configurations are set using the SageMaker Python SDK Estimator, and which is fit using the training data from the Processing Job that was run earlier.

```
[ ]: estimator = sagemaker.estimator.Estimator(
    container,
    iam_role,
    hyperparameters=hyperparameters,
    instance_count=1,
    instance_type="ml.m5.2xlarge",
)
estimator.fit({"train": train_input})
```

Now that you have a trained model there are a number of different things you can do. For more details on training with SageMaker, please see <https://sagemaker.readth>

3. You can monitor the status of submitted training job in SageMaker Console under Training / Training jobs tab on the left.



4. Host trained Model for real time inference

Deploy model for real-time inference

1. We will now use another notebook provided under project folder `hosting/Model_deployment_Steps.ipynb`. This is a simple notebook with 2 cells - First cell has code for deploying your model to persistent endpoint. Here you need to update `model_url` with your training job output S3 model artifact. Here are image for reference.

Amazon SageMaker Model Deployment using persistent endpoint for Real-Time inference

In this notebook we will deploy the Model that was trained using the preprocessed training_input from Data Wrangler preprocessing Job.

Perform model deployment using Amazon SageMaker

```
[ ]: import boto3
import sagemaker
import time
from sagemaker import get_execution_role, session
from time import gmtime, strftime
from sagemaker.model import Model
from sagemaker.image_uris import retrieve
from sagemaker.predictor import Predictor
from sagemaker.serializers import CSVSerializer

# Setup defaults
region = boto3.Session().region_name
role = get_execution_role()

# Update this model_url with the trained model generated from the training job that we ran as part of Data Wrangler
model_url = '<<Your S3 location of pre-trained model artifact>>'

# Pull the Amazon SageMaker built-in algorithm for xgboost
image_uri = retrieve('xgboost', region, '1.2-1')
# Create a model object using Model class
model = Model(image_uri=image_uri, model_data=model_url, role=role)

endpoint_name = 'DEMO-xgb-readmission-prediction-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print("EndpointName={}".format(endpoint_name))

# Deploy the model as an endpoint hosted by Amazon SageMaker
hosted_endpoint = model.deploy(initial_instance_count=1,
                               instance_type='ml.m4.xlarge',
                               endpoint_name=endpoint_name)
```

2. The second cell in the notebook will run inference on sample test file test_data_UCI_sample.csv .

Perform inference against the deployed model

```
[ ]: # Create SageMaker Predictor object to pass in the test data for inference
predictor = Predictor(endpoint_name=endpoint_name, serializer=CSVSerializer())

print("Sending test traffic to the endpoint {}. \nPlease wait...".format(endpoint_name))

with open("test_data/test_data_UCI_sample.csv", 'r') as f:
    for row in f:
        payload = row.rstrip('\n')
        print('Incoming Payload => ', payload)
        response = predictor.predict(data=payload).decode('utf-8')
        print("Inference Output => ", response)
        time.sleep(1)

print("Done!")
```

Clean up

After you have experimented above steps, perform the below 2 clean-up steps to stop incurring charges.

1. Delete hosted endpoint. You can do this from within SageMaker Console as shown below.

The screenshot shows the Amazon SageMaker console's 'Endpoints' page. The left sidebar contains navigation links: Amazon SageMaker Studio, Dashboard, Search, Images, Ground Truth, Notebook, Processing, Training, Inference, Compilation jobs, Model packages, Models, Endpoint configurations, and Endpoints. The main content area shows a table of endpoints. The first endpoint, 'DEMO-xgb-readmission-prediction-2021-08-30-16-43-41', is highlighted with a red box. Above the table, there are buttons for 'Update endpoint', 'Actions', 'Add/Edit tags', 'Delete', and 'Create endpoint'. A red arrow labeled '1' points to the 'Actions' button, and a red arrow labeled '2' points to the 'Delete' option in the dropdown menu.

Name	ARN	Creation time	Status	Last updated
DEMO-xgb-readmission-prediction-2021-08-30-16-43-41	arn:aws:sagemaker:us-east-2:2021-08-30-16-43-41	Aug 30, 2021 16:43 UTC	InService	Aug 30, 2021 16:53 UTC
huggingface-finetune-2021-08-25-23-29-16	arn:aws:sagemaker:us-east-2:2021-08-25-23-29-16	Aug 25, 2021 23:29 UTC	InService	Aug 25, 2021 23:35 UTC
reorder-classifier-2021-07-13-21-59-48-953	arn:aws:sagemaker:us-east-2:2021-07-13-21-59-48-953	Jul 13, 2021 21:59 UTC	InService	Jul 31, 2021 11:36 UTC

2. Shutdown Data Wrangler App. You can do this from within SageMaker Console by navigating to your SageMaker user-profile - as shown below.

Amazon SageMaker > SageMaker Studio > Control Panel

User Details

User summary

Edit user

Delete user

Open Studio

User profile name

sm-user

Status

Ready

Created

Tue Jul 13 2021 13:38:20 GMT-0400 (Eastern Daylight Time)

Studio ID

Execution role

arn:aws:iam:::role/service-role/AmazonSageMaker-ExecutionRole-20210206T181522

Apps

App name	Status	App type	Created	
sagemaker-debugger-1-ml-m5-4xlarge-4061c3f60cc59b594d8d07fc0603	Deleted	KernelGateway	Wed Aug 25 2021 18:57:25 GMT-0400 (Eastern Daylight Time)	Delete app
datascience-1-0-ml-t3-medium-1abf3407f667f989be9d86559395	Ready	KernelGateway	Wed Aug 25 2021 17:19:29 GMT-0400 (Eastern Daylight Time)	Delete app
sagemaker-data-wrang-ml-m5-4xlarge-b741c1a025d542c78bb538373f2d	Ready	KernelGateway	Wed Jul 14 2021 12:59:38 GMT-0400 (Eastern Daylight Time)	Delete app
default	Ready	JupyterServer	Tue Jul 13 2021 13:39:18 GMT-0400 (Eastern Daylight Time)	Delete app

SageMaker Projects and JumpStart

Conclusion

This concludes the example. In this example you have learnt how to use SageMaker Data Wrangler capability to create data preprocessing, feature engineering steps using simple to use Data Wrangler GUI. We then used the generated notebook to submit a SageMaker managed processing job to perform the data preparation using our data flow file. Later we saw how to train a simple XGBoost algorithm using our processed dataset. In the end we hosted our trained model and ran inferences against synthetic test data.