Option 1: Amazon SageMaker Data Wrangler and Feature Store

Overview

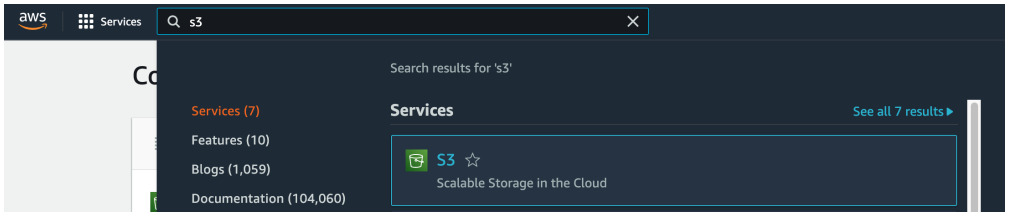
In this lab, you will learn how to use Amazon SageMaker Data Wrangler to prepare data for machine learning. You will also use Amazon SageMaker Feature to store, retrieve, and share machine learning (ML) features.

Dataset upload to S3 bucket

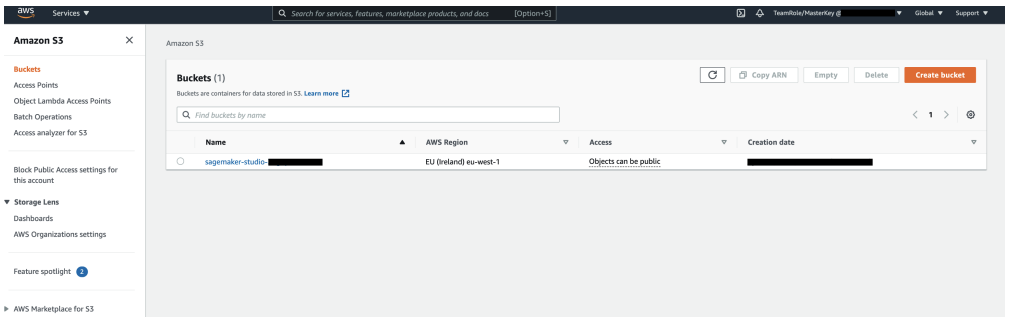
1. Type the following URL in your browser to download the dataset that we are going to use in the lab:

```
https://sagemaker-sample-data-us-west-2.s3-us-west-2.amazonaws.com/autopilot/direct_marketing/bank-
```

2. Unzip it on your computer.
3. Go to the AWS Management Console, search **S3** in the searchbox on top of your console, then go to **S3** service console.

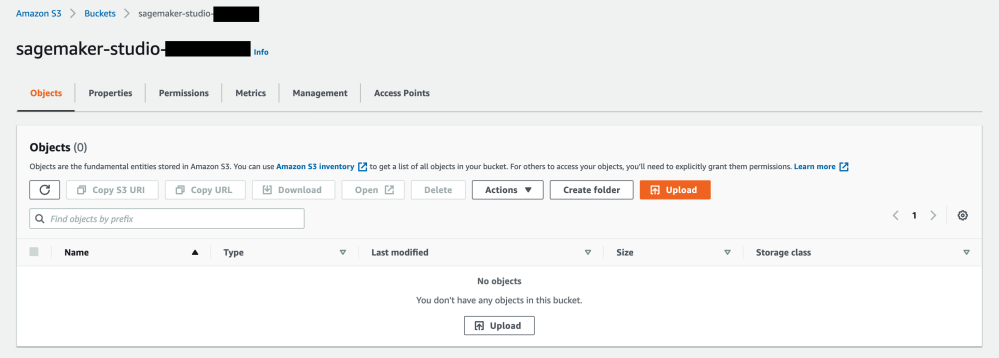


4. In the S3 console, click on the **sagemaker-studio-*** bucket.

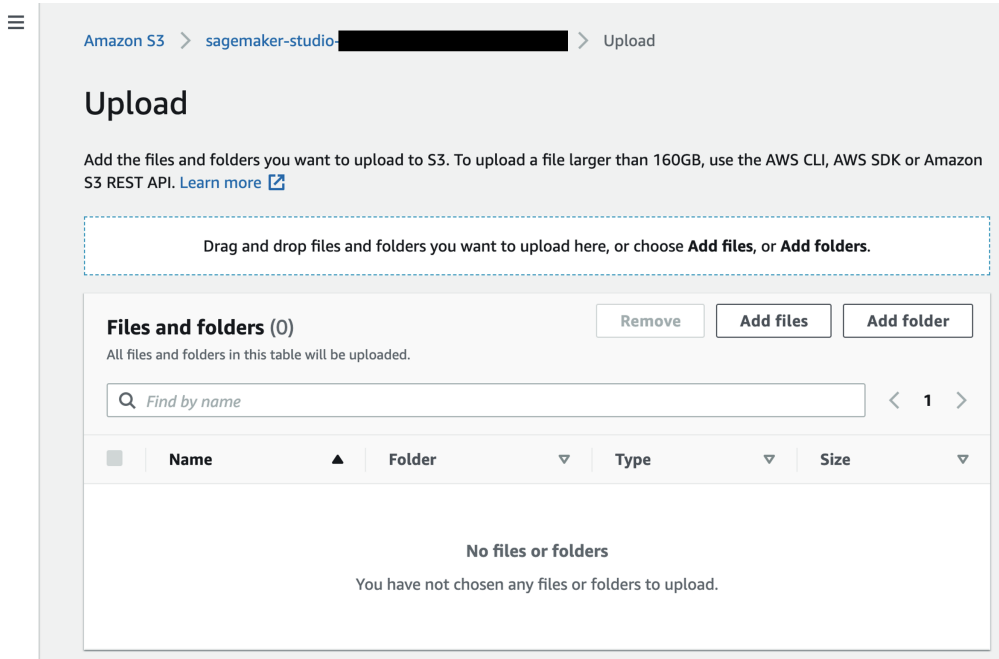


> The **sagemaker-studio-*** bucket was created automatically when you created the SageMaker Studio domain in the **Prerequisites** section. If you follow the **Event Engine** track, the bucket was preprovisioned by you instructor.

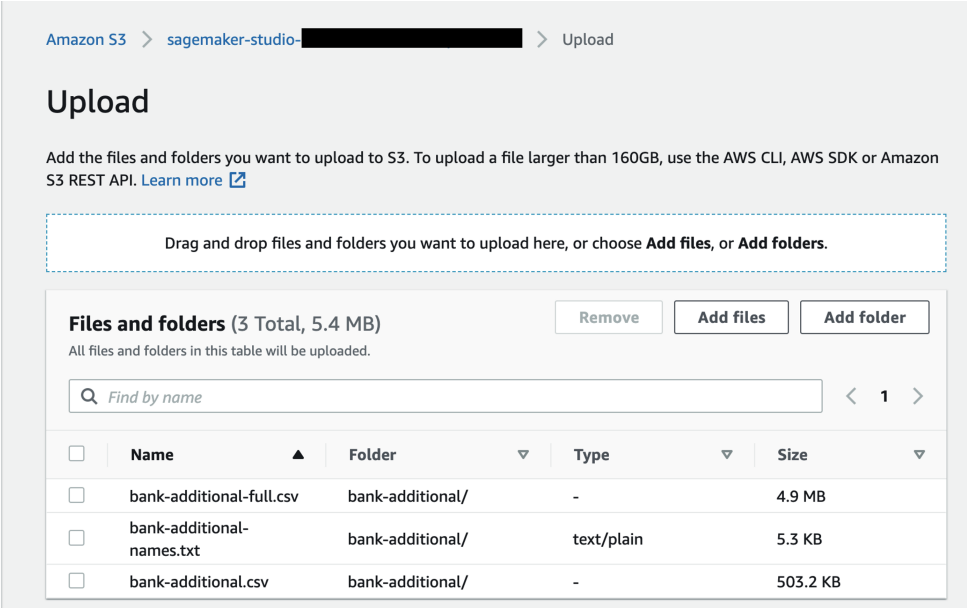
5. Click **Upload**.



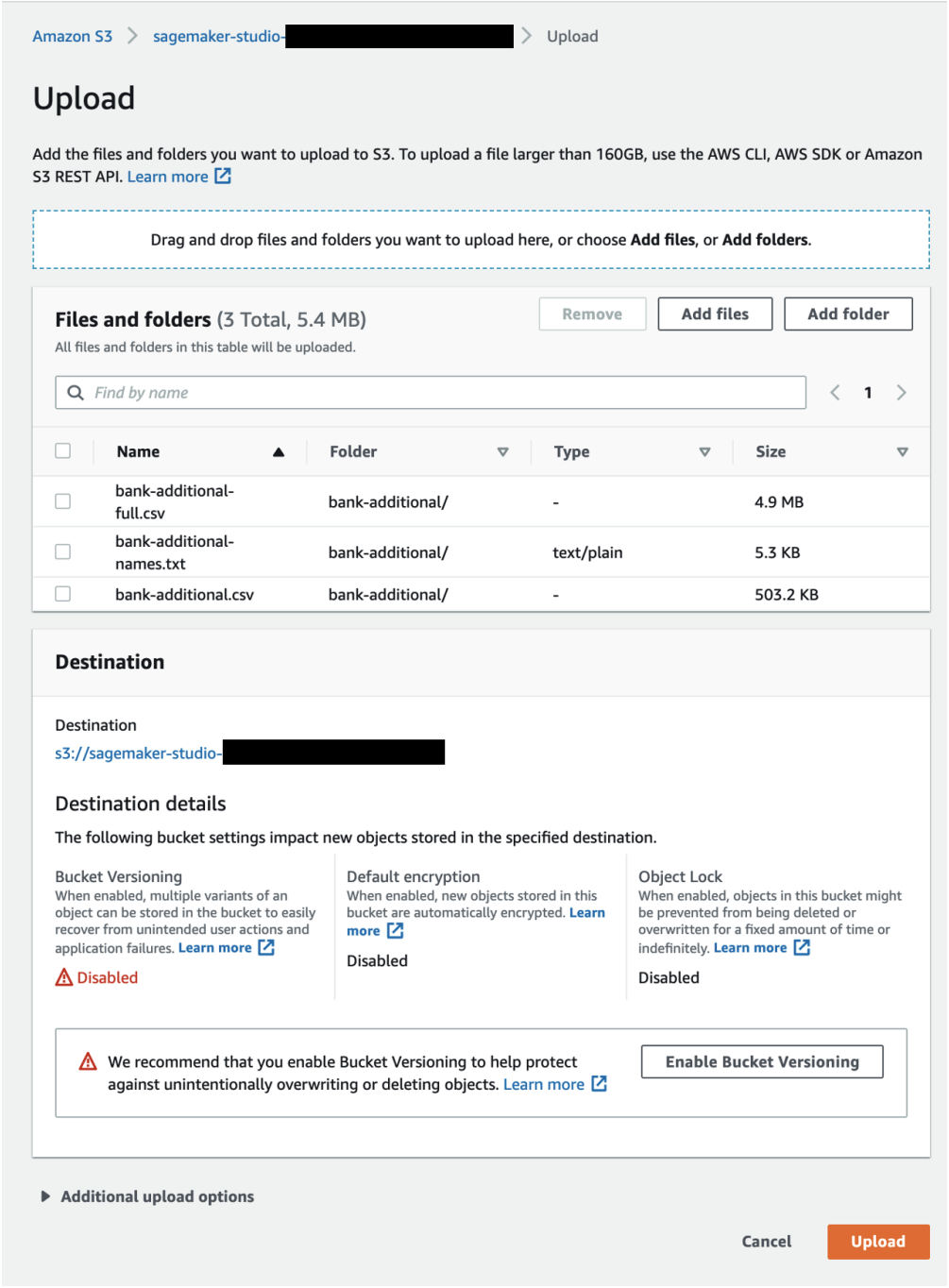
6. On the Upload page, drag and drop the unzipped **bank-additional** folder into the drag and drop area.



The 3 files inside the bank-additional folder will appear in the **Files and folders** area.

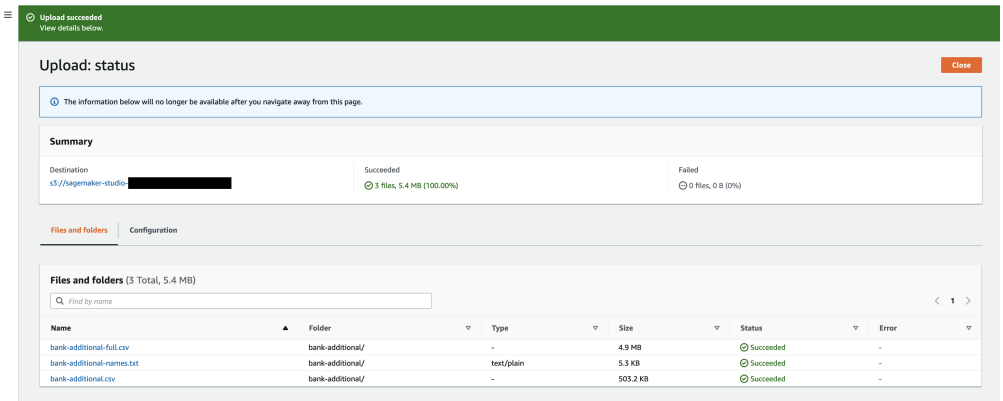


7. Click **Upload** at the bottom of the page.

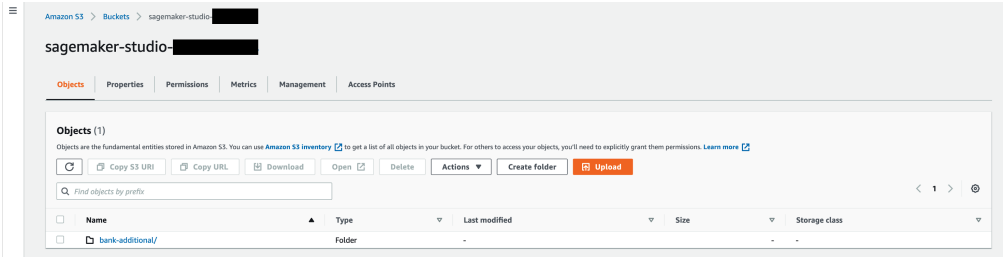


Once the upload is finished, you will see a green band with a message **Upload succeeded**.

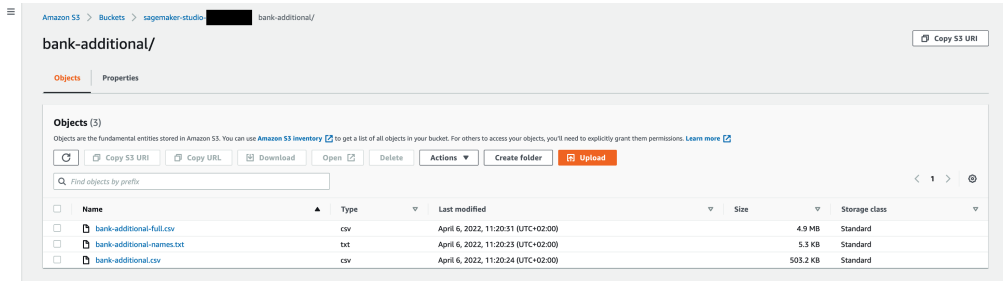
8. Click on **Close**.



You can verify that the folder is present in the S3 bucket.



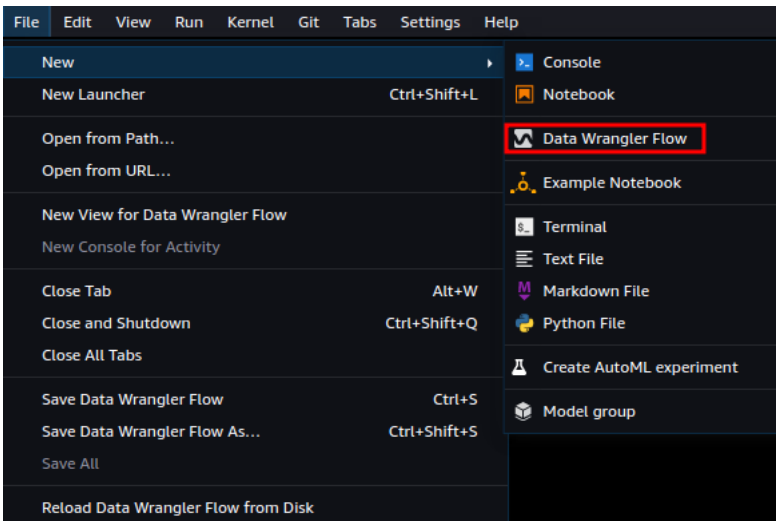
You will see the 3 files if you click on the **bank-additional** folder in the S3 bucket.



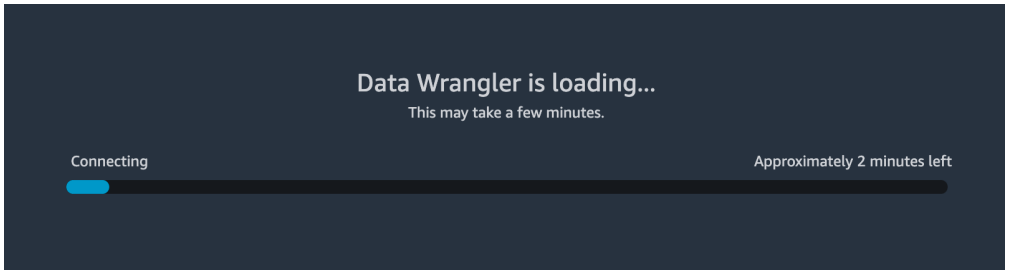
Congratulations!! You have successfully uploaded the dataset to Amazon S3.

Amazon SageMaker Data Wrangler initial setup

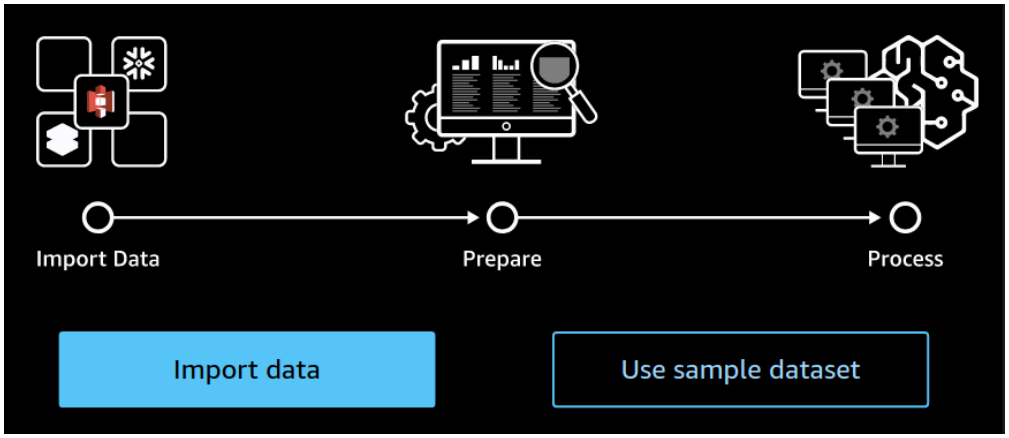
1. Go back to the SageMaker Studio Notebook that you created in the **Prerequisites** section.
2. From the top menu, select **File** → **New** → **Data Wrangler Flow**



An instance will be launched for the newly created Data Wrangler workflow, which may take a few minutes.



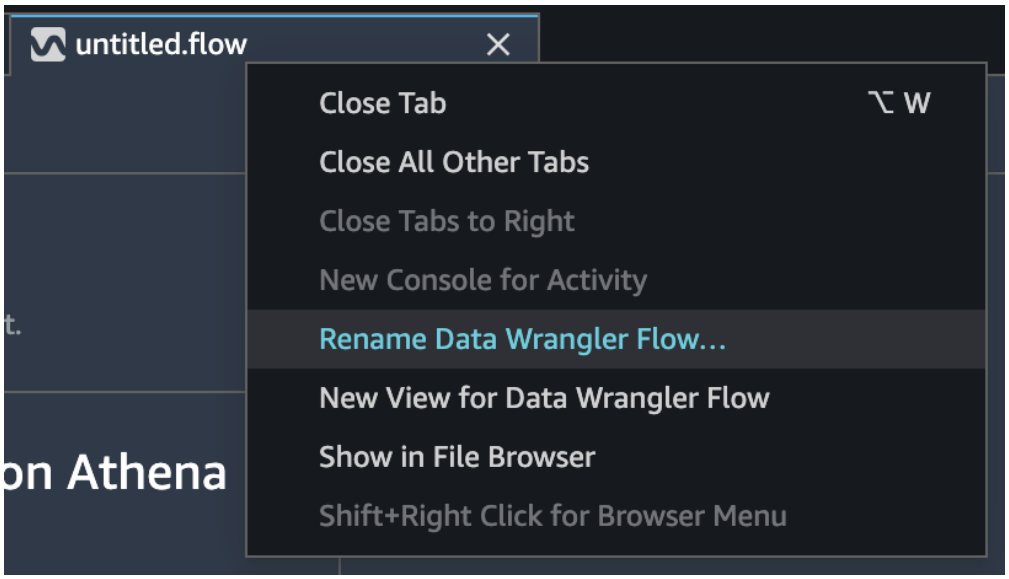
After a while, the loading message disappears, data source screen is shown:



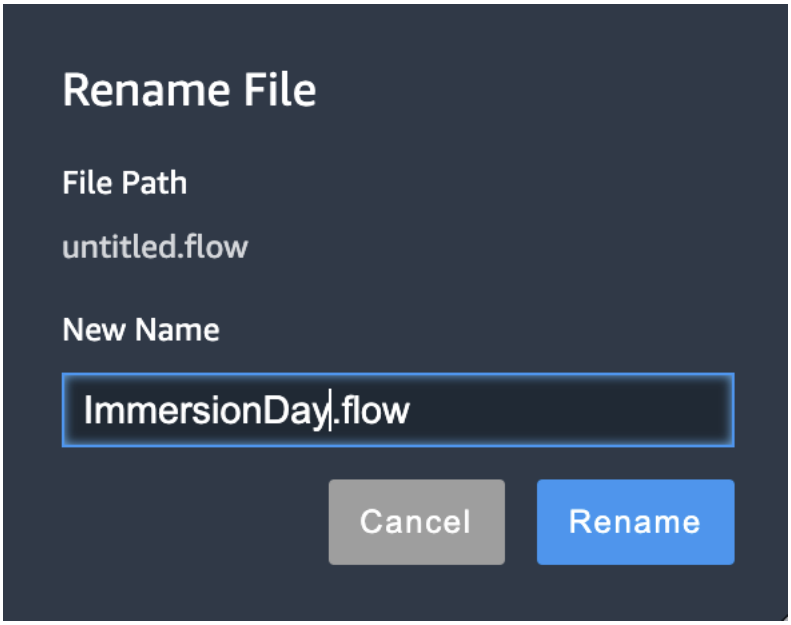
Add a S3 data file as Data Wrangler source

To use Data Wrangler, you need access to a m5.4xlarge Amazon Elastic Compute Cloud (Amazon EC2) instance in your account. If the AWS account you use has been provided by an instructor, you will have access to this instance type.

Right click on the file tab to rename it to **ImmersionDay.flow.**

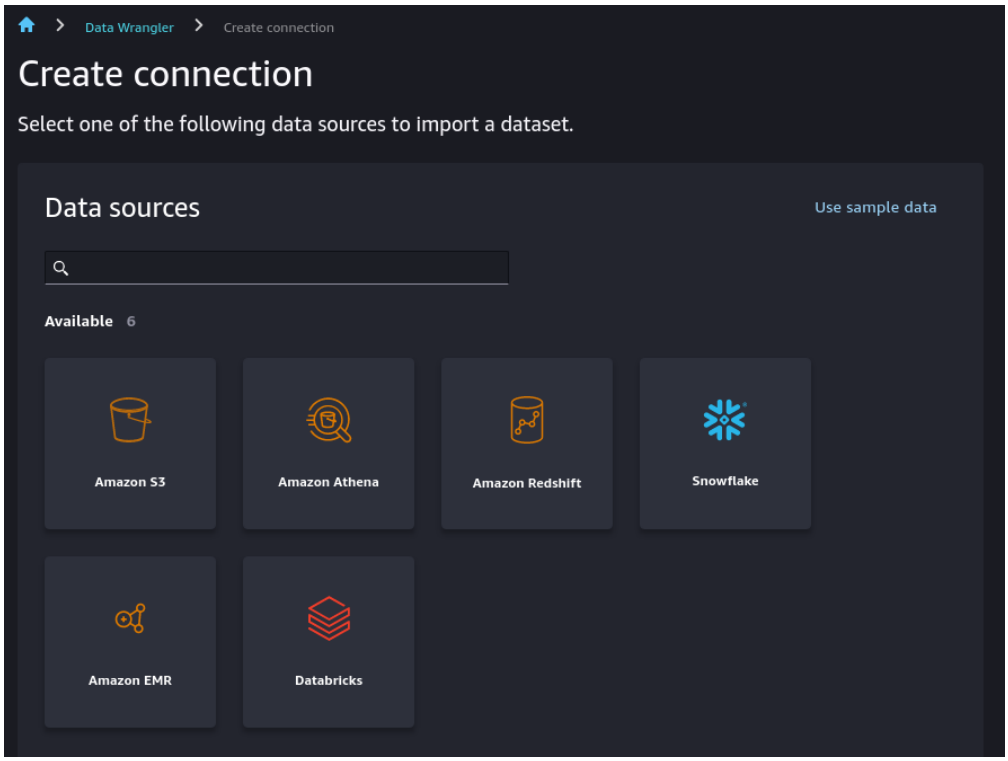


Then click the Rename button.



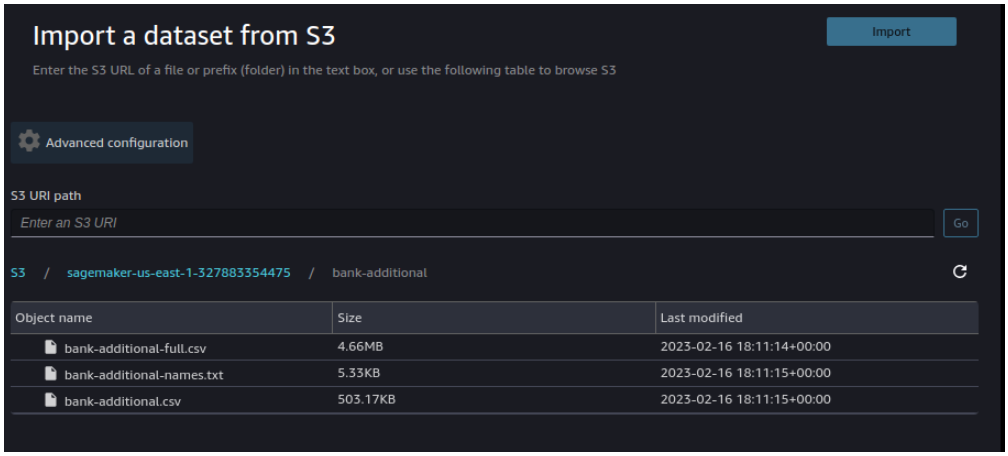
Click **Import data**, then click **Amazon S3**.

Double-click on the **sagemaker-*** bucket.

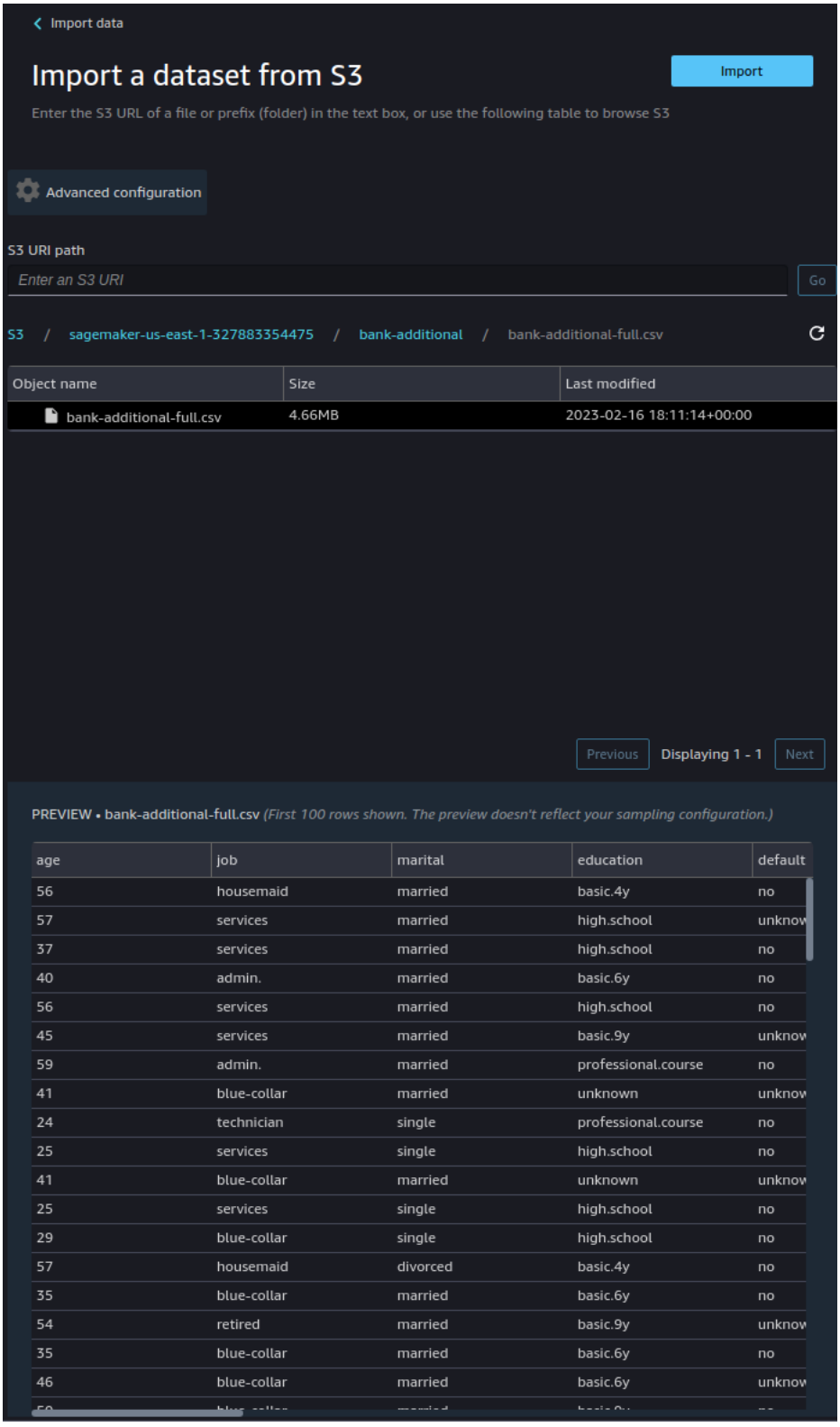


Double-click on the **bank-additional** folder to open it.

Click on the **bank-additional-full.csv** file.

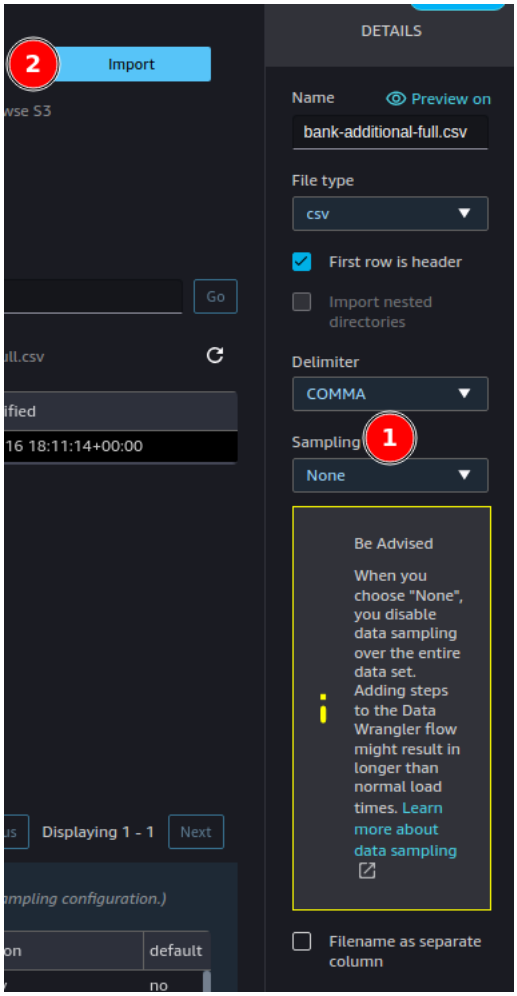


A preview of the data is displayed:

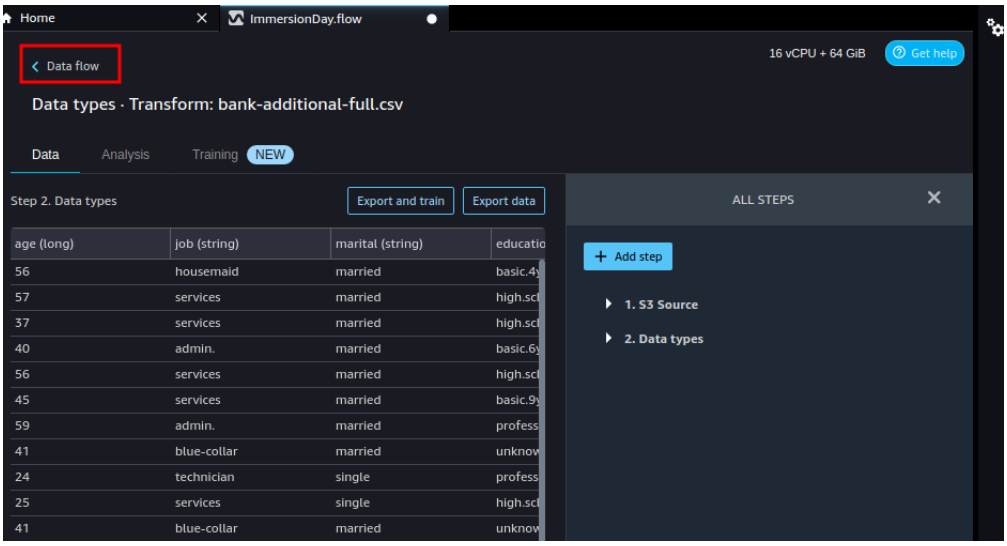


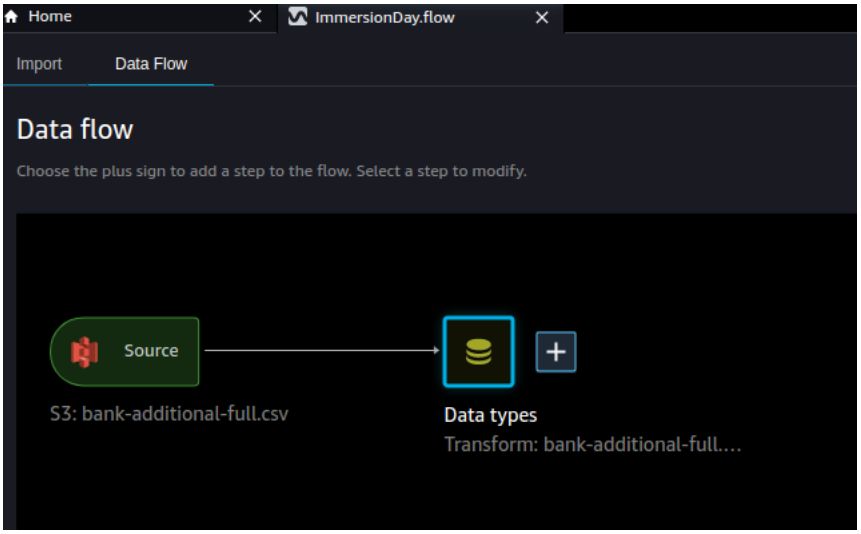
For the **Sampling** option select **None.** This is a relatively small dataset, no need to sample.

Click on **Import dataset.**



The **Data types** screen will appear. Click the arrow at the top next to **Data Flow** to see the new flow file.

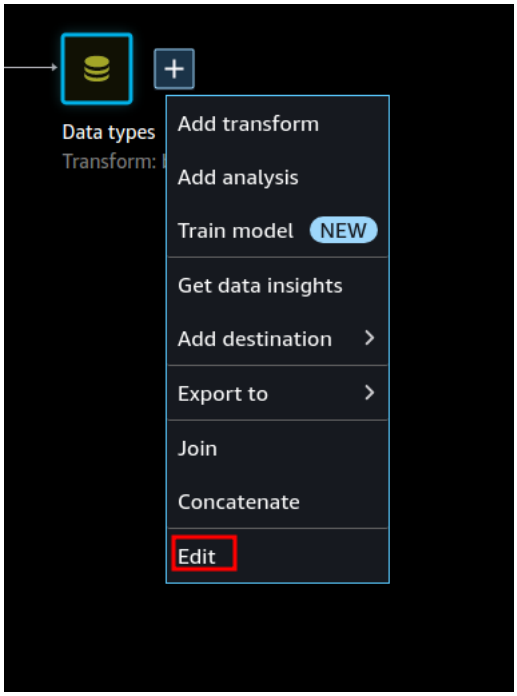




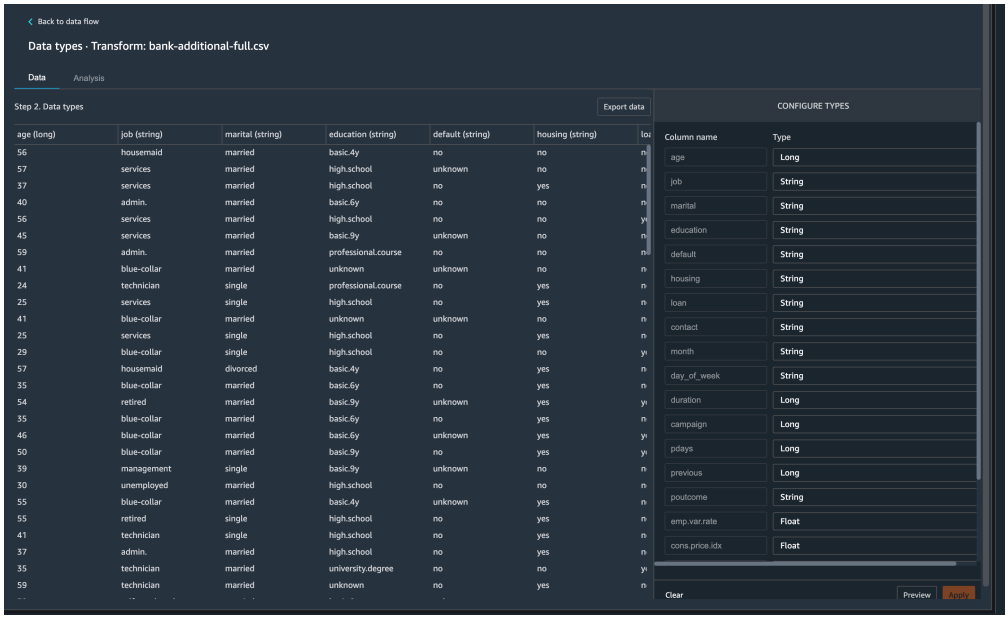
Dataset analysis

In this section, we are going to analyze the dataset.

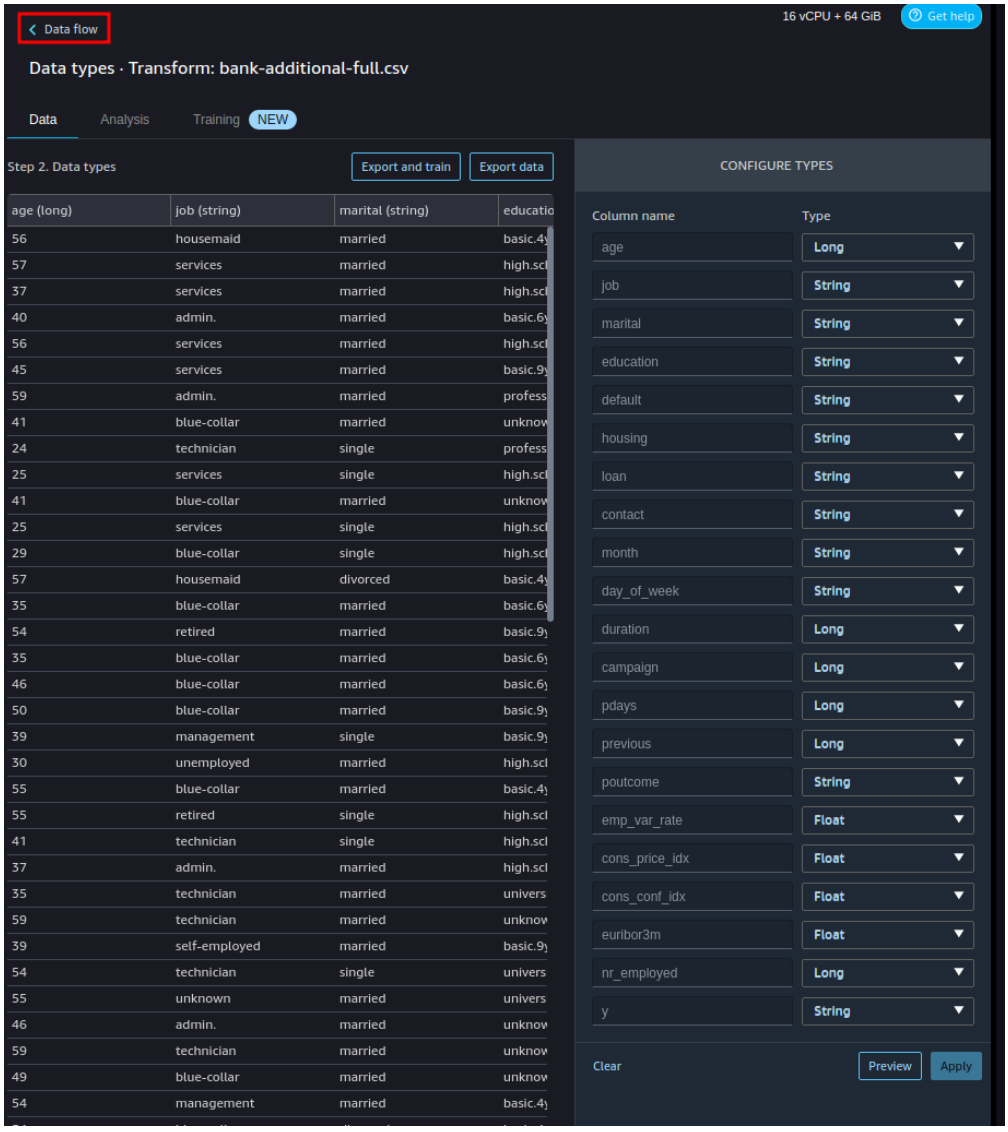
To get an overview of the data types, click on the **+** sign next to the **Data Types** node, and select **Edit:**



You get an overview of the dataset with the column names and types:



We don't want to change any type, so we click on **< Data Flow**:



The specifics on each feature are the following: **Demographics:**

- **age**: Customer's age (numeric)

- **job**: Type of job (categorical: 'admin.', 'services', ...)
- **marital**: Marital status (categorical: 'married', 'single', ...)
- **education**: Level of education (categorical: 'basic.4y', 'high.school', ...)

Past customer events:

- **default**: Has credit in default? (categorical: 'no', 'unknown', ...)
- **housing**: Has housing loan? (categorical: 'no', 'yes', ...)
- **loan**: Has personal loan? (categorical: 'no', 'yes', ...)

Past direct marketing contacts:

- **contact**: Contact communication type (categorical: 'cellular', 'telephone', ...)
- **month**: Last contact month of year (categorical: 'may', 'nov', ...)
- **day_of_week**: Last contact day of the week (categorical: 'mon', 'fri', ...)
- **duration**: Last contact duration, in seconds (numeric). Important note: If duration = 0 then y = 'no'.

Campaign information:

- **campaign**: Number of contacts performed during this campaign and for this client (numeric, includes last contact)
- **pdays**: Number of days that passed by after the client was last contacted from a previous campaign (numeric)
- **previous**: Number of contacts performed before this campaign and for this client (numeric)
- **poutcome**: Outcome of the previous marketing campaign (categorical: 'nonexistent','success', ...)

External environment factors:

- **emp.var.rate**: Employment variation rate - quarterly indicator (numeric)
- **cons.price.idx**: Consumer price index - monthly indicator (numeric)
- **cons.conf.idx**: Consumer confidence index - monthly indicator (numeric)
- **euribor3m**: Euribor 3 month rate - daily indicator (numeric)
- **nr.employed**: Number of employees - quarterly indicator (numeric)

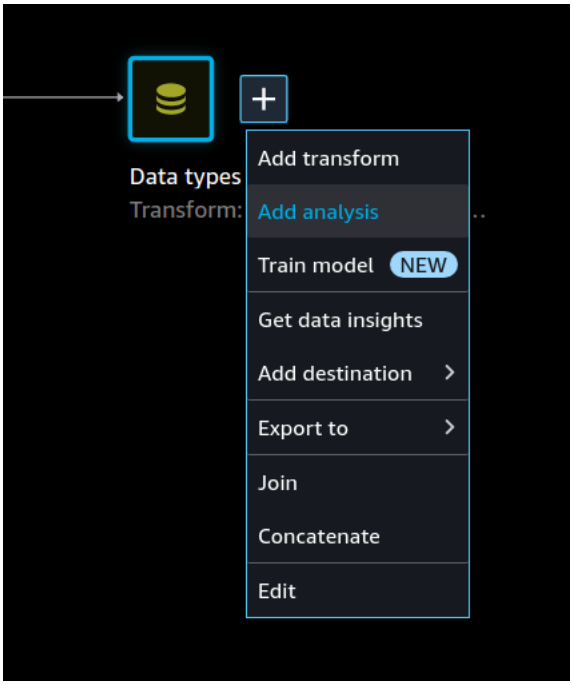
Target variable:

- **y**: Has the client subscribed a term deposit? (binary: 'yes','no')

Data Quality and Insights Report

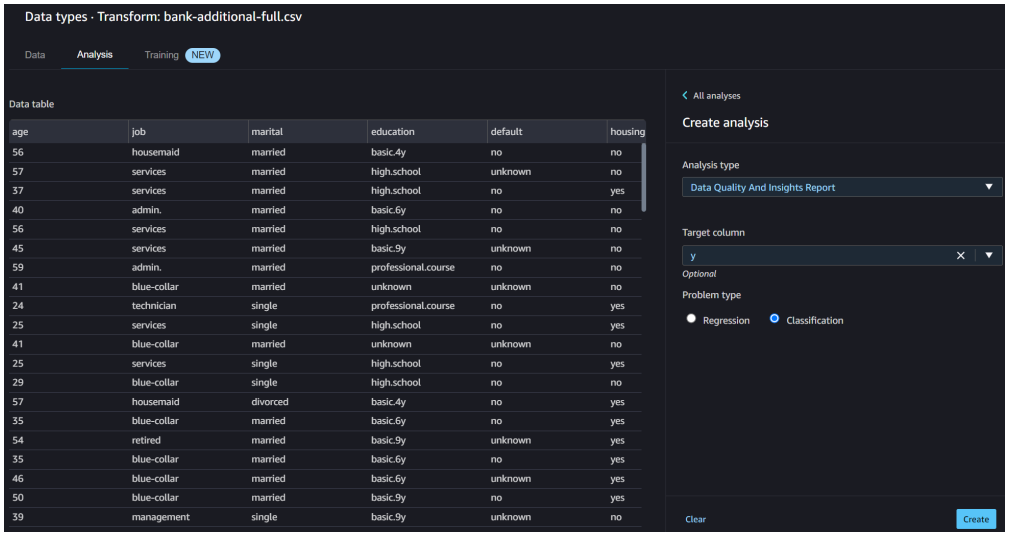
Data Wrangler's built-in **Data Quality and insights Report** provides and automated Exploratory Data Analysis (EDA) report on your imported dataset. Included in this report is feature summary, a quick model review, and much more information that would help in data processing and cleaning.

Click on the **+** sign again and then on **Add analysis**:



You are presented with the **Analysis** screen.

1. Under **Analysis type**, select **Data Quality and Insights Report**, and for **Target column**, select **y**.
2. Select **Classification** for **Problem type.**
3. Click **Create.**



After some time, the **Data Quality and Insights Report** content will populate. Scroll down the generated report to see all populated content. Lets review some of the topics in these report.

- **Data Statistics**
  - Our dataset contains 41188 rows and 21 features.
  - No missing rows
  - A few duplicate rows. We can drop duplicate rows using the **Drop duplicates** built-in transform.
  - Breakdown of the type of features

Dataset statistics

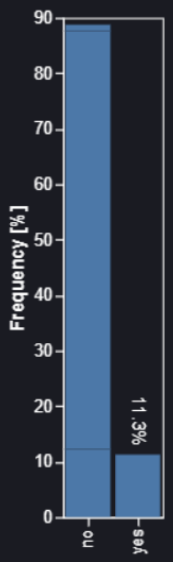
Key	Value
Number of features	21
Number of rows	41188
Missing	0%
Valid	100%
Duplicate rows	0.0583%

Feature type	Count
numeric	10
categorical	9
text	0
datetime	0
binary	1
unknown	0

- **Target Column**
  - This report clearly shows the imbalance in our dataset target label.
  - `no` class compromises of ~88% of the entire data.
  - This imbalanced data will pose problems during model training. One way to remediate imbalaced data is to use the built-in **Balance data** transform.

TARGET COLUMN

key	value
Number of classes	2
Valid	100%
Missing	0%



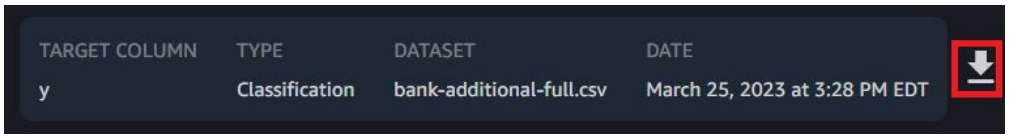
Histogram of the frequent values of the target column.

- **Quick model**
  - Our dataset achieves a validation accuracy of 91.7% and a decent balanced accuracy score of 75.4%
  - Minority class, `yes`, performs poorly on all mtrics including Recall, Precision and F1-Score.
  - Quick model metrics will serve as a benchmark for any training job down the line to compare model perfromance.

Metric	Validation scores	Train scores
Accuracy	0.917	0.932
Balanced accuracy	0.754	0.792
ROC-AUC	0.95	0.965
F1	0.596	0.669
Precision	0.661	0.739
Recall	0.543	0.612

class	precision	recall	f1-score	support
no	0.9432775919732441	0.9645690834473324	0.953804531619885	7310.0
yes	0.6605504587155964	0.5431034482758621	0.5960969840331165	928.0

To download this report, click on the download icon on the upper-right corner of the report page.

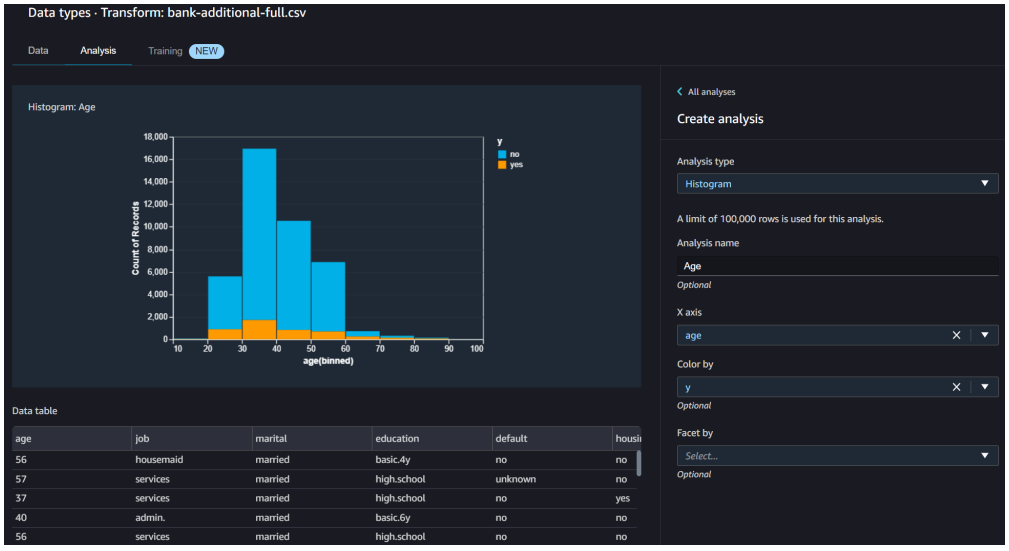


Now that we've viewed a good general overview of the data, lets create a second analysis to analyze the correlation between the target variable, and other variables.

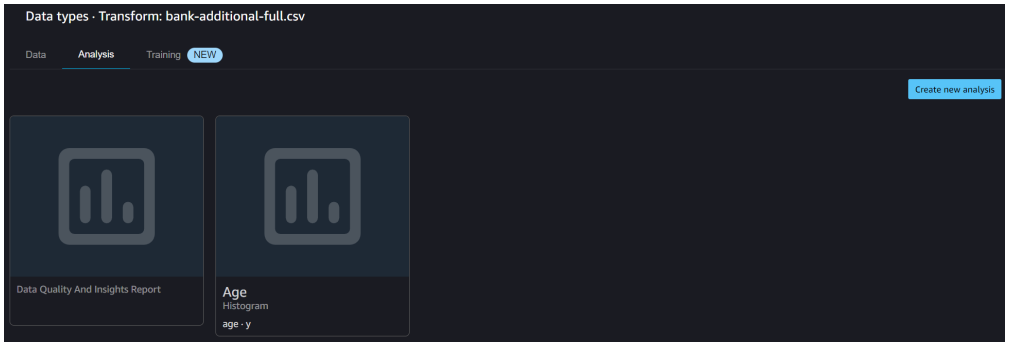
Histogram

1. Create a new Analysis as you did before.
2. Under the **Analysis type** drop-down, select **Histogram**
3. Enter **Age** for **Analysis name**
4. Choose `age` as **X-axis** and **Color by** `y` as shown below and click on **Preview**.

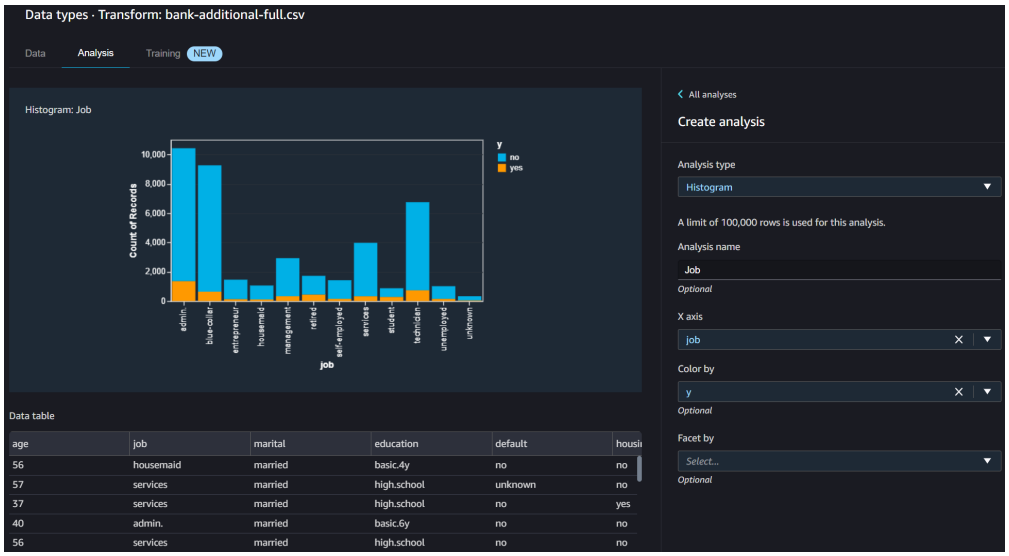
You will see the following histogram and will have an visual representation of the potential influence of the `age` feature of a person on the target variable `y`, which represents if the person will accept a marketing offer:



Once you click on **Save**, you will be able to access your analysis from the ***Transform*** screen **Analysis** tab:



You can repeat the operation for job, marital status and even other features to visualize the influence on the target variable.



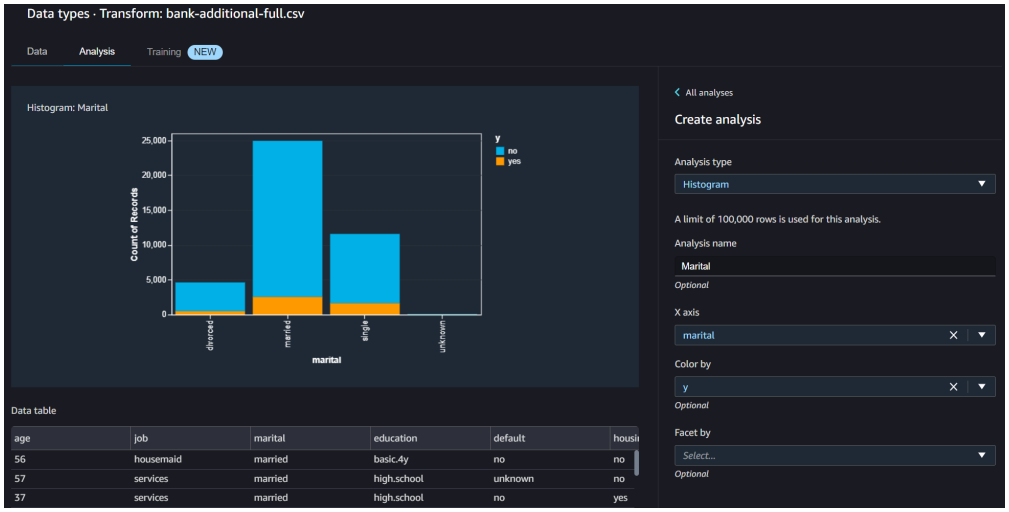
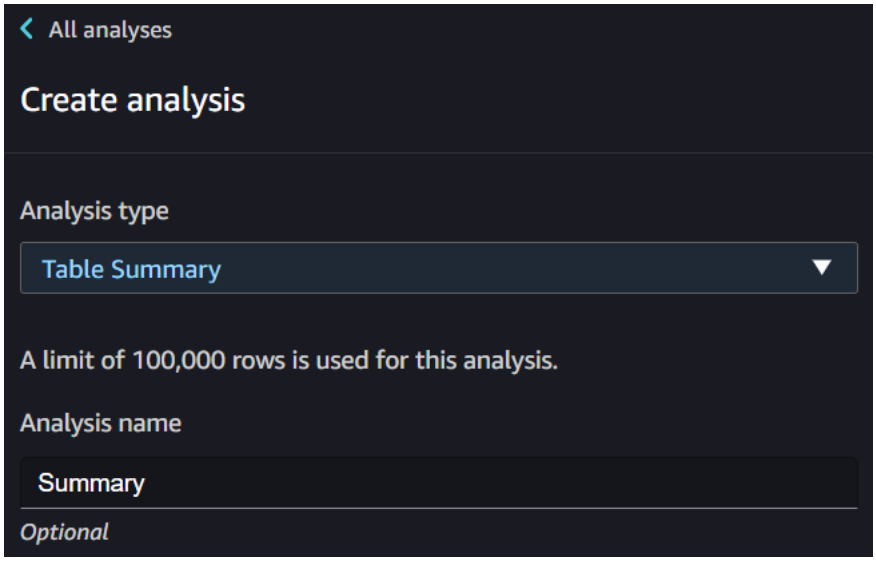


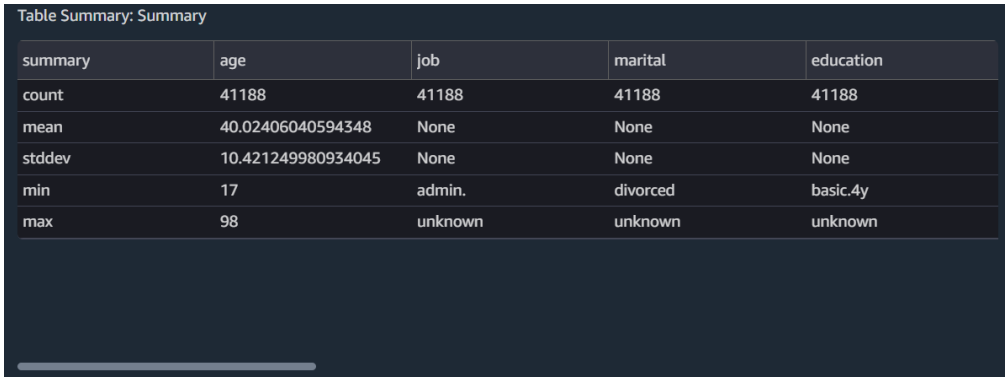
Table Summary

For a more general table statistics overview, we can run a **Table Summary** analysis.

1. Create another Analysis.
2. Under **Analysis type**, select **Table Summary**, and give the analysis a name (e.g. **Summary**).



3. Click on **Preview**:



summary	age	job	marital	education
count	41188	41188	41188	41188
mean	40.02406040594348	None	None	None
stddev	10.421249980934045	None	None	None
min	17	admin.	divorced	basic.4y
max	98	unknown	unknown	unknown

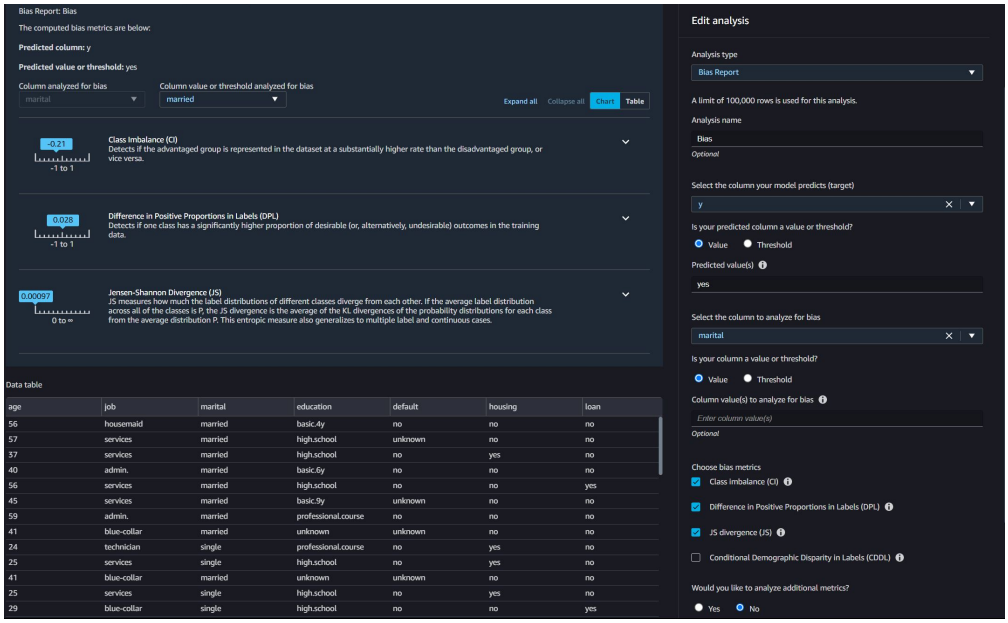
A table summary appears. Click on **Save** to save your analysis.

In addition to the dataset feature and row count, you have a view on different statistics of the dataset (e.g. mean, standard deviation). Once saved, you can access your analysis from the **Transform** screen **Analysis** tab.

Bias Report

Next, lets gain some insight into any potential bias at the different `marital` groups by creating a **Bias Report**.

1. Create a new Analysis.
2. For Analysis Type, select Bias Report.
3. For name, enter *Bias*.
4. For target, select `y`.
5. For predicted column value or threshold, select **Value**.
6. For predicted value(s) enter `yes`.
7. For Column to analyze for bias, enter `marital`.
8. For Column value(s) to analyze for bias, leave empty. This will provide bias for all the values in the `marital` facet.
9. For Bias metrics, check all **EXCEPT CDDL**.
10. Select **No** for analyzing additional metrics.
11. Click **Check For Bias**.



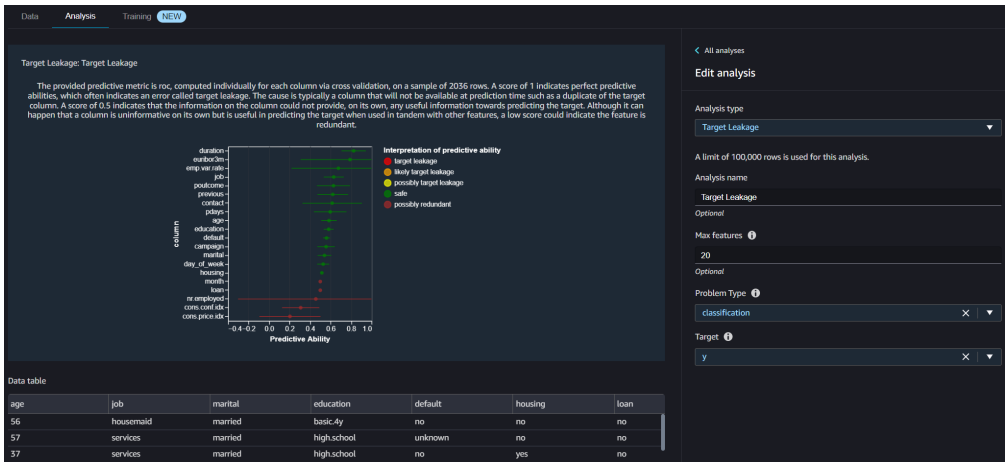
From the results, we observe a class imbalance where the `married` class is 21% more represented than other classes. We also observe that the `married` class is 2.8% less likely to subscribe to a bank term deposit. For more information on all the bias metrics supported by Data Wrangler, learn how Amazon SageMaker Clarify helps detect bias.

12. Click **Save**.

Target Leakage

For our final analysis, lets create a **Target Leakage** analysis. Target leakage occurs when there is data in a machine learning training dataset that is strongly correlated with the target label, but is not available in real-world data.

1. Create a new Analysis.
2. For **Analysis Type**, select **Target Leakage**.
3. For name, enter *Target Leakage*.
4. For **Problem Type**, select **Classification.**
5. For **Target**, select `y`.
6. Click **Preview**.

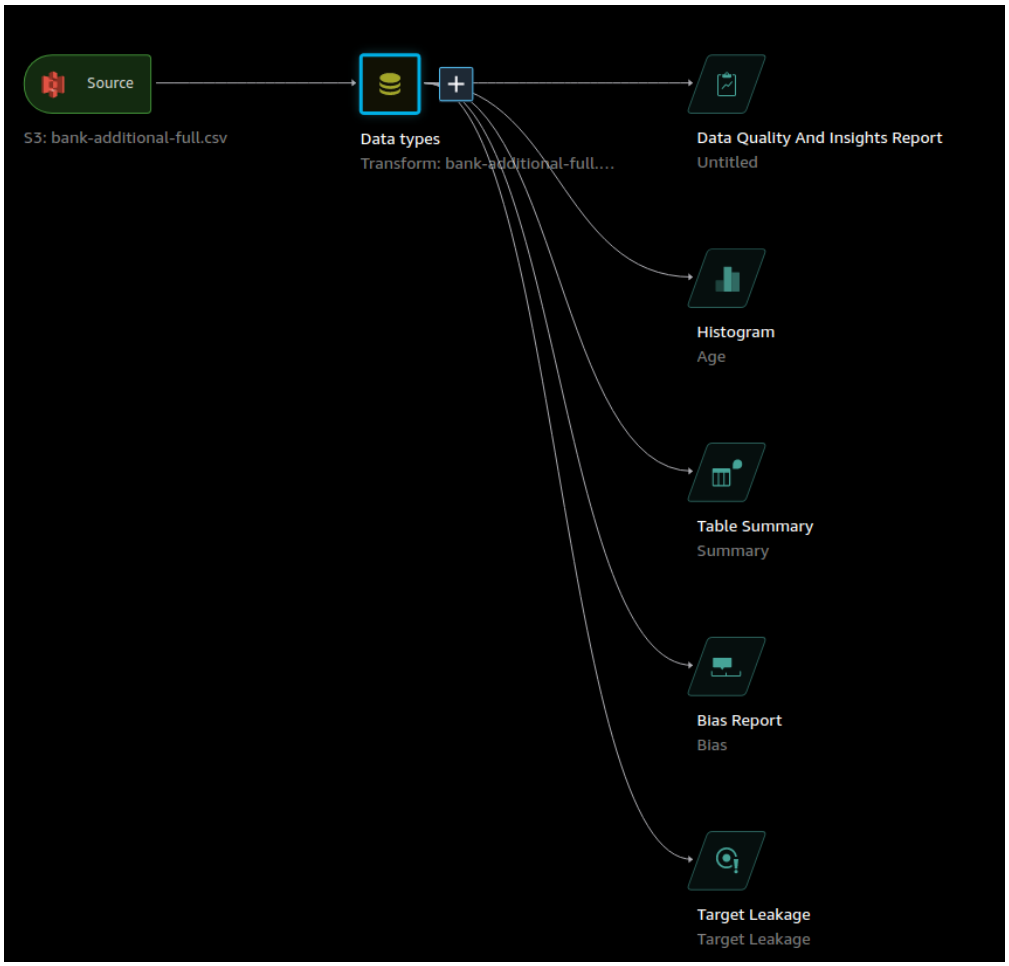


From the result, we see that the `cons.conf.idx`, `cons.price.idx`, `nr.employed`, `loan` and `month` features have low predictive abilities and are possibly redundant features. We would drop some of this features later on.

7. Click **Save**.

Data Wrangler samples your data to perform Target Leakage analysis, therefore, you may see discrepancies in the charts for each successive analysis.

At this point, your flow should look similar to the image below:



We've performed an import, and analyzed our data. Now we can take some actions based on our analyses and domain knowledge to transform the data to make it read for training.

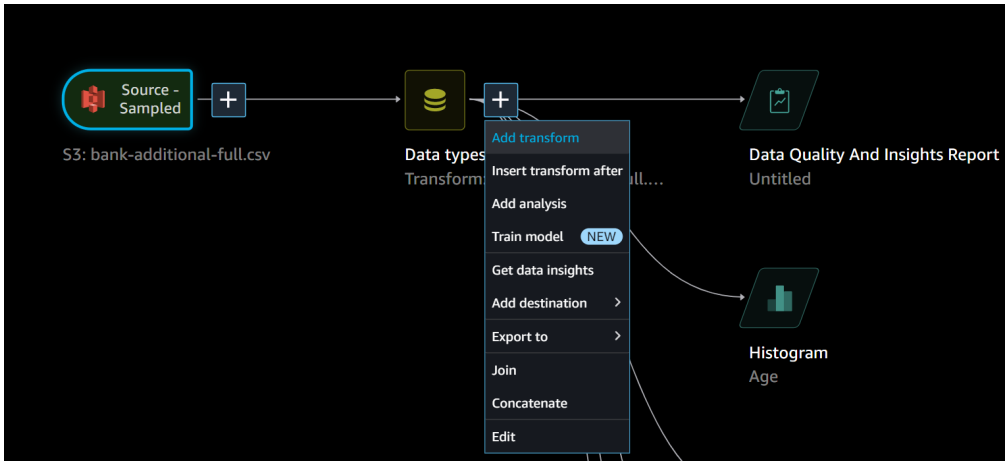
Data Transformation

Cleaning up data is part of nearly every machine learning project. It arguably presents the biggest risk if done incorrectly and is one of the more subjective aspects in the process.

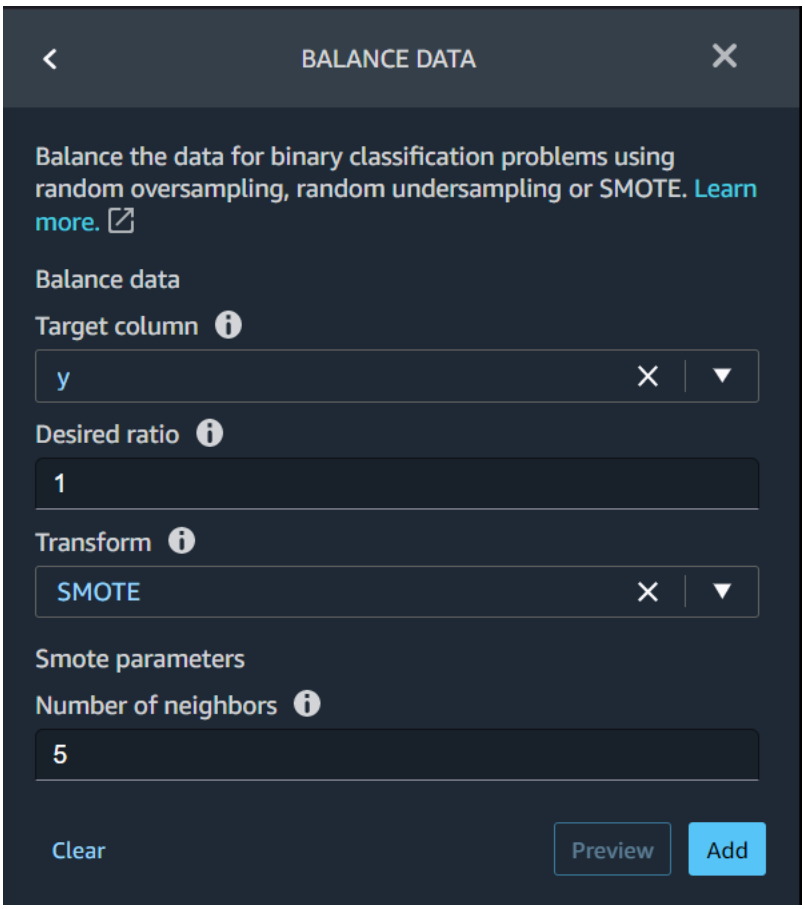
Handle Class Imbalance

From the Data Quality and Insights Report above, we discovered an imbalanced representation of the target class `y`. Let's make use of the **Synthetic Minority Over-sampling Technique (SMOTE)** to balance our data target class representation.

In order to add transformation, go to the **Data flow** tab and click on the **+** sign next to **Data types** and select **Add transform**.



1. Choose **+ Add step** -> **Balance data** transform
2. Choose **SMOTE** as the transform type.
3. Select y as **Target column** and keep rest parameters as default.
4. Click **Preview** and **Add**.



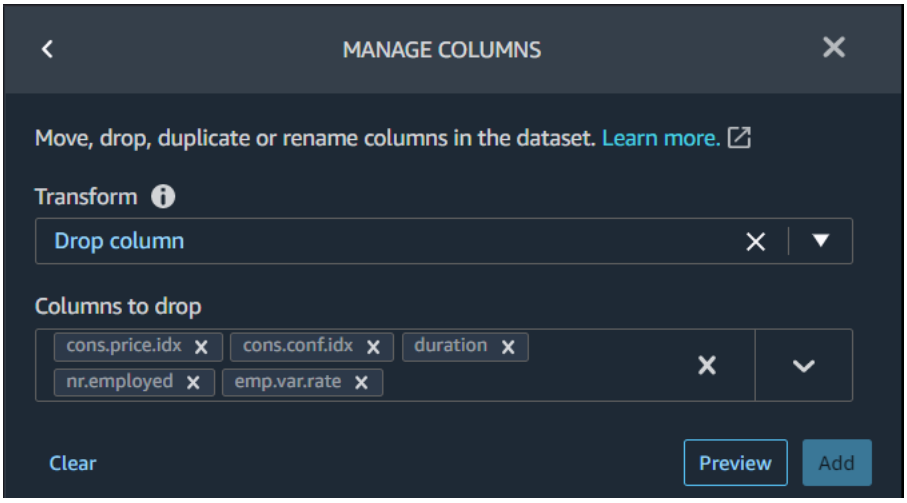
Drop Columns

We would remove certain features from our dataset.

1. Under **ALL STEPS**, click **+ Add Step**.
2. Click on **Manage columns**. `(Can also search for Manage columns in the search bar instead of scrolling)`
3. Select **Drop column** and choose the following:
   - **duration**
   - **emp.var.rate**

- **cons.price.idx**
- **cons.conf.idx**
- **nr.employed**

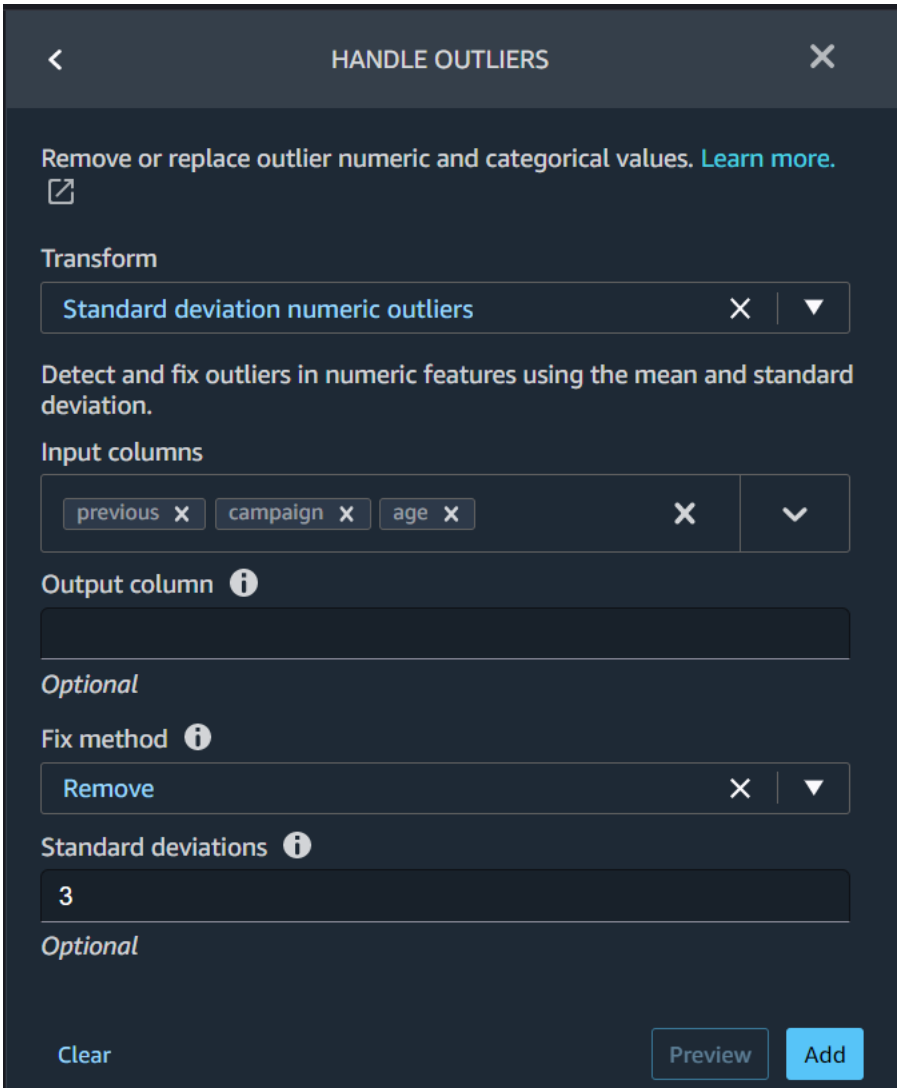
4. choose **Preview** and **Add**



Handle Feature Outliers

We also noticed, from the Data Quality and Insights Report above, some anomalous data samples which may be a result of outliers in our data. Let's remove outliers from numerical features in our dataset.

1. Under **ALL STEPS**, click **+ Add Step**.
2. Click on **Handle Outliers**. `(Can use the search bar)`
3. Select **Standard deviation numeric outliers** under **Transforms** and choose the following under **Input columns**:
   - **previous**
   - **age**
   - **campaign**
4. Type 3 for **Standard deviations** (the criteria of outliers).
5. choose **Preview** and **Add**

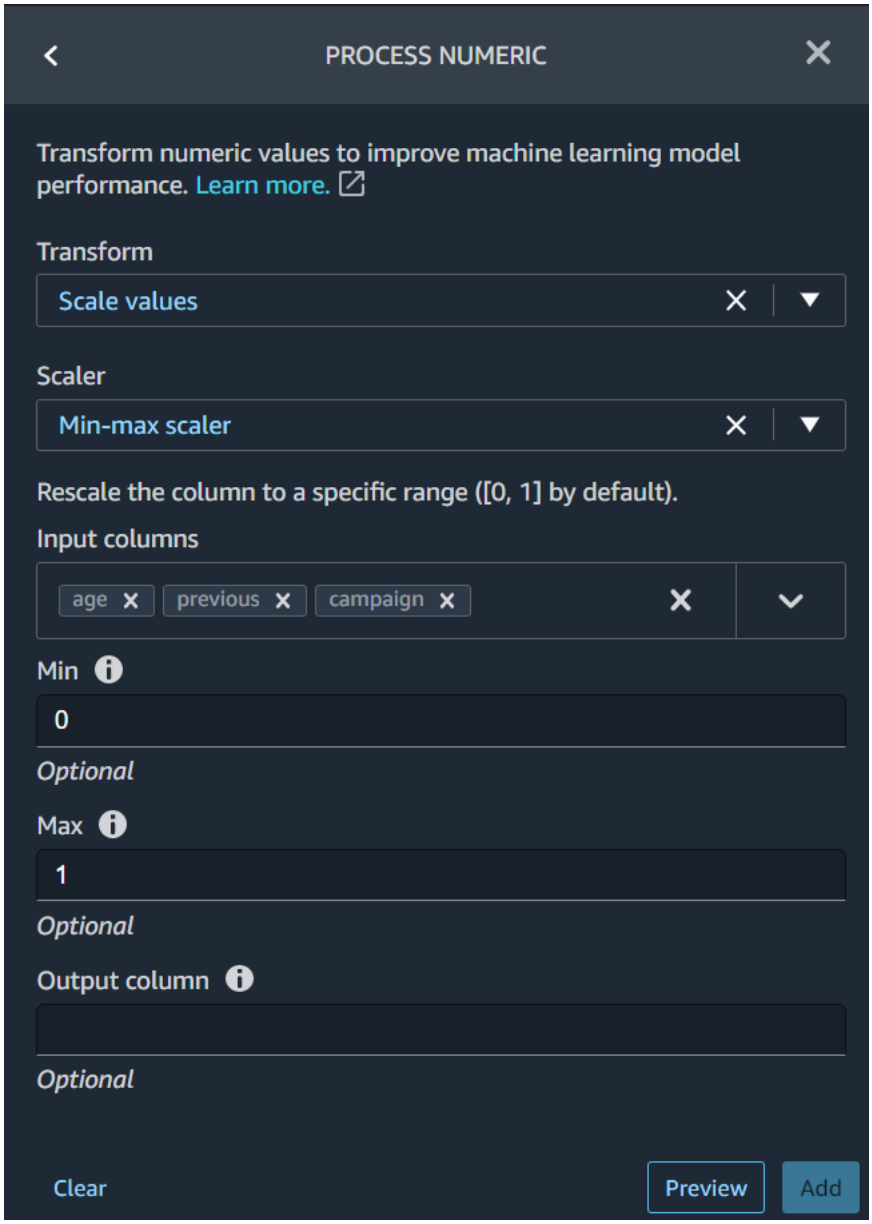


Scale Numerical Features

Scaling is required to rescale the data and it's used when we want features to be compared on the same scale for our algorithm. And, when all features are in the same scale, it also helps algorithms to understand the relative relationship better and converge faster.

Let's use min-max scaler to scale all numeric columns to values between 0 and 1.

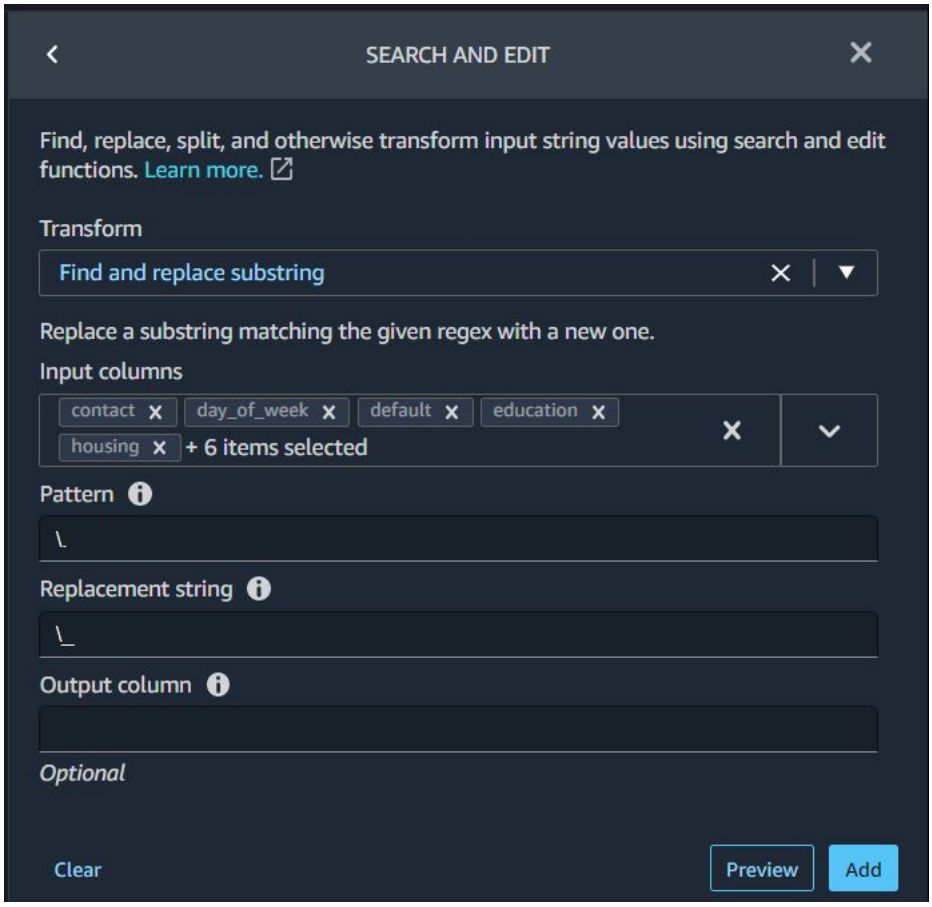
1. Under **ALL STEPS**, click **+ Add Step**.
2. Click on **Process Numeric**. `(Can use the search bar)`
3. Select **Min-max scaler** under **Scaler** and choose the following under **Input columns**:
   - **previous**
   - **age**
   - **campaign**
4. choose **Preview** and **Add**



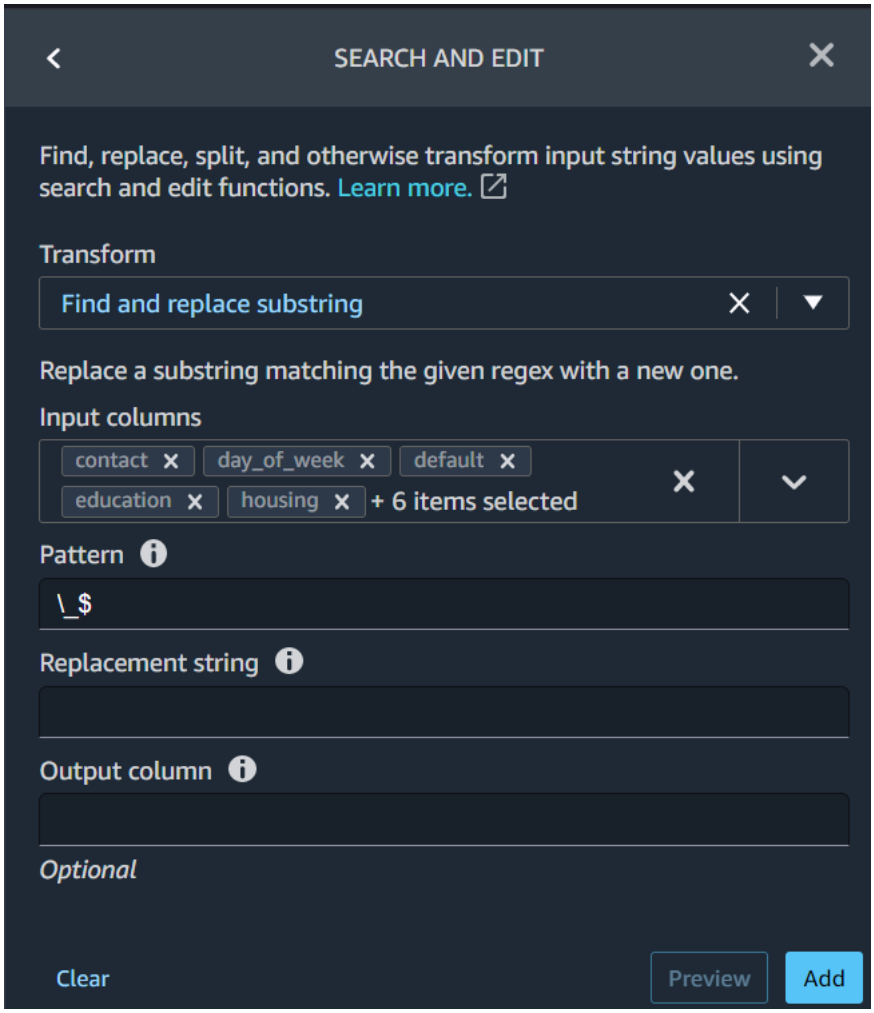
Replace Feature Characters

We would replce the dot `.` values into `_` values for all our categorical features. This is important when exporting data to Feature Store, as feature names with `.` are not currently supported by Feature Store.

1. Under **ALL STEPS**, click **+ Add Step**.
2. Click on **Search and edit**. `(Can use the search bar)`
3. Select **Find and replace substring** under **Transform** and choose all string type columns under **Input columns**
4. For **Pattern**, type in `\.` and `\_` for **Replacement string**
5. choose **Preview** and **Add**



We also notice that certain feature values end with a `_` , lets remove this appendix. Repeat the same process above, however, input `\_$` for **Symbols** and leave **Replacement string** empty (put an S and delete it to be able to proceed with this field empty).



Custom Transforms

The Custom Transforms step allows you to define custom transformations in **Python (User-Defined Function)**, **Python (PySpark)**, **Python (Pandas)**, and/or **SQL (PySpark SQL)**. We will add a custom transformation step to do the following.

- Add an indicator variable to capture when pdays takes a value of 999.
- Add an indicator for individuals not actively employed
- Add unique ID and date for features store
- Compute the Variance Inflation factor (VIF) of various numerical features in the dataset and drop columns with a VIF value greater than 1.2. This helps us mitigate any multicolinearity in our dataset.

Follow the step below to implement this custom transformation in either **Python (Pandas)** or **Python (PySpark)** by choosing the appropiate tab.

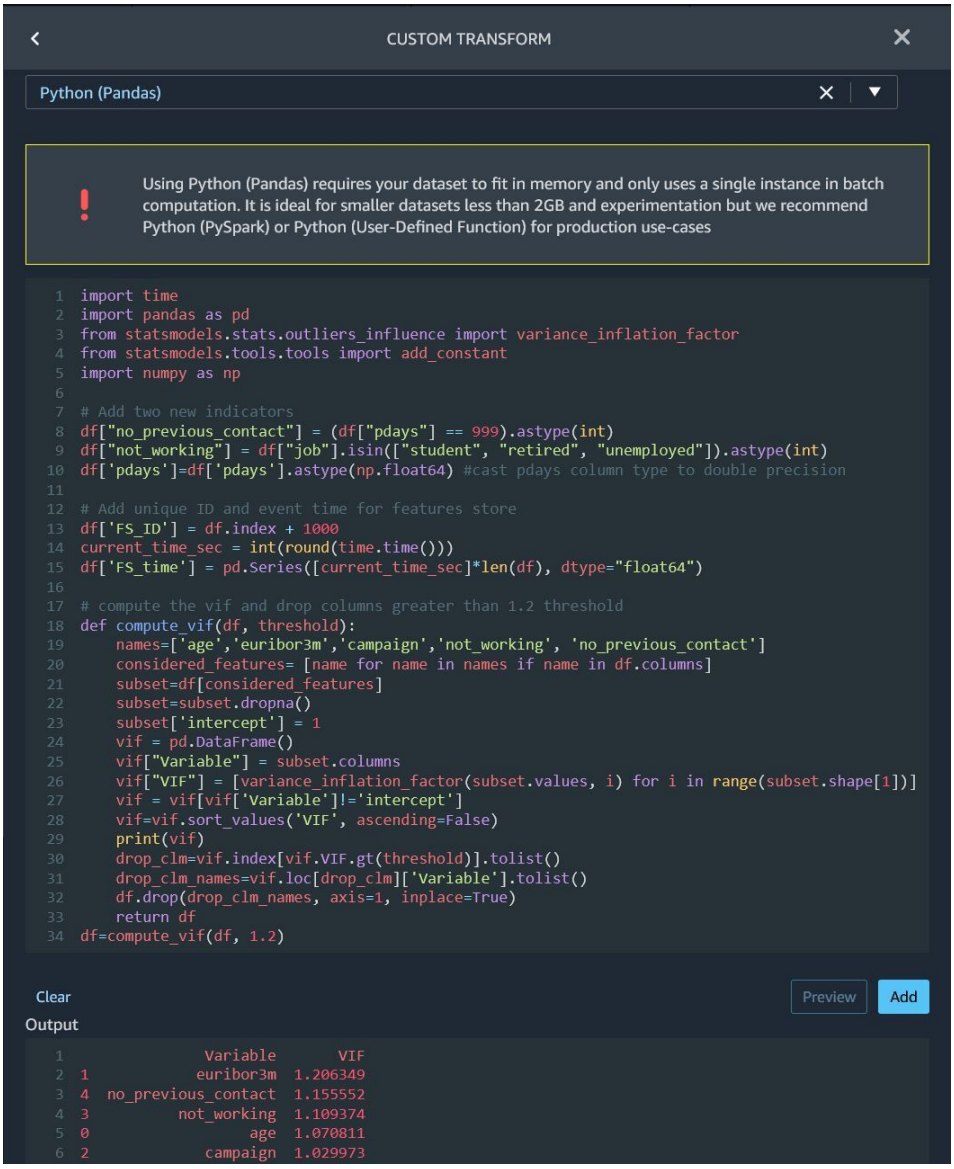
Python (Pandas) | Python (PySpark)

Under **ALL STEPS** on the right select **+ Add Step** then **Custom Transform**. Change to **Python (Pandas)**. You will get a warning, which is not a problem. Copy and paste this script:

```
import time
import pandas as pd
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
import numpy as np

# Add two new indicators
df["no_previous_contact"] = (df["pdays"] == 999).astype(int)
```

```
df["not_working"] = df["job"].isin(["student", "retired", "unemployed"]).astype(int)
df['pdays']=df['pdays'].astype(np.float64) #cast pdays column type to double precision

# Add unique ID and event time for features store
df['FS_ID'] = df.index + 1000
current_time_sec = int(round(time.time()))
df['FS_time'] = pd.Series([current_time_sec]*len(df), dtype="float64")

# compute the vif and drop columns greater than 1.2 threshold
def compute_vif(df, threshold):
    names=['age','euribor3m','campaign','not_working', 'no_previous_contact']
    considered_features= [name for name in names if name in df.columns]
    subset=df[considered_features]
    subset=subset.dropna()
    subset['intercept'] = 1
    vif = pd.DataFrame()
    vif["Variable"] = subset.columns
    vif["VIF"] = [variance_inflation_factor(subset.values, i) for i in range(subset.shape[1]
    vif = vif[vif['Variable']!='intercept']
    vif=vif.sort_values('VIF', ascending=False)
    print(vif)
    drop_clm=vif.index[vif.VIF.gt(threshold)].tolist()
    drop_clm_names=vif.loc[drop_clm]['Variable'].tolist()
    df.drop(drop_clm_names, axis=1, inplace=True)
    return df
df=compute_vif(df, 1.2)
```

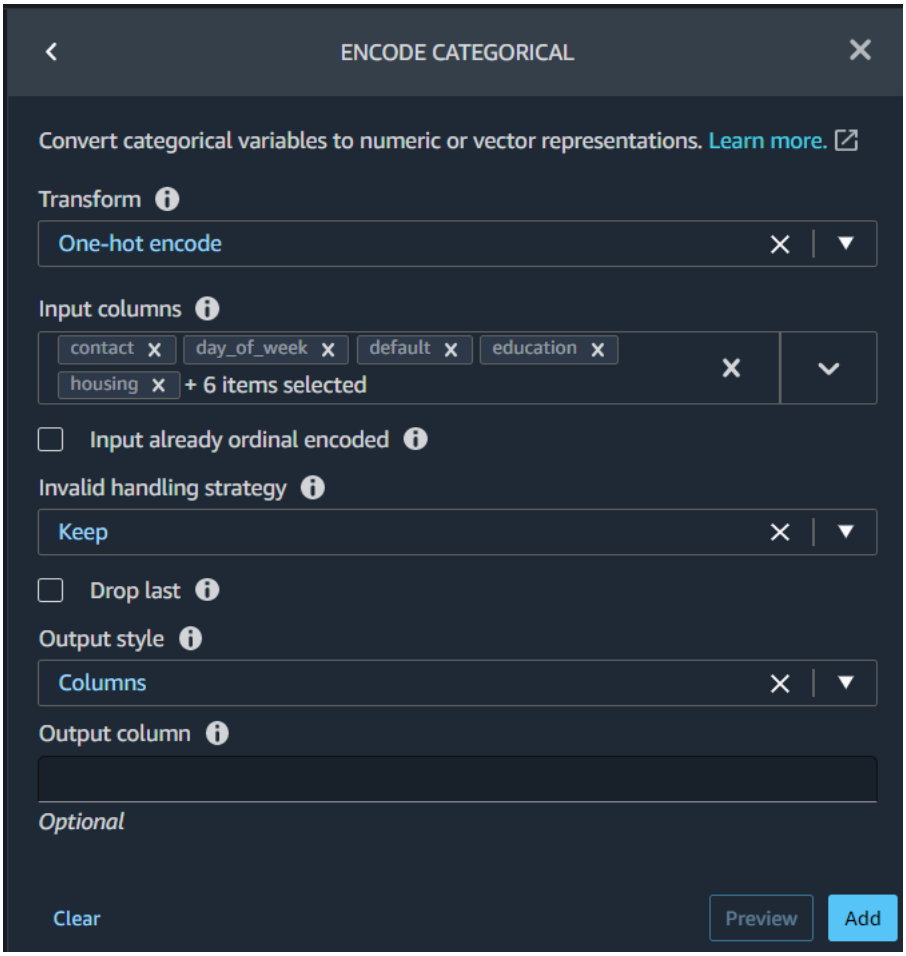
Choose **Preview**, then choose **Add**.



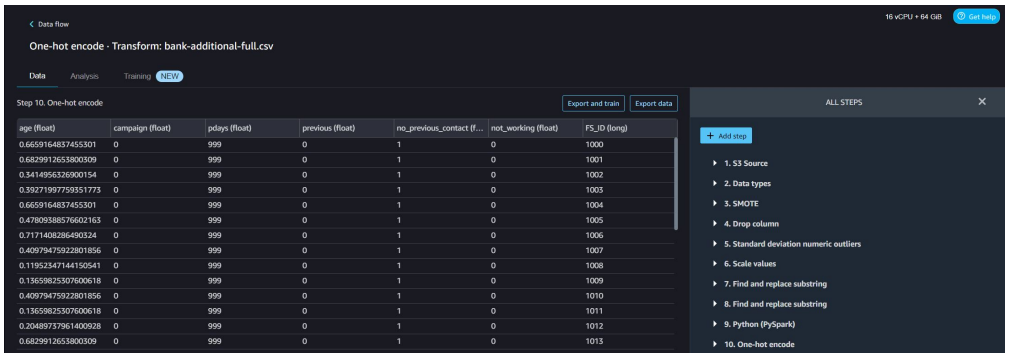
One-hot encoding the categorical variables

To convert categorical variables into sets of indicators, we use one hot encoding. This will encode the categorical features as a one-hot numeric array.

1. Click **Add Step**
2. Choose **Encode Categorical**.
3. Select **One-hot encode** as **Transform** and choose all string type columns as **Input columns**.
4. Make sure select **Columns** as **Output style**.
5. Click on **Preview**, then **Add** to add the transform to the data flow.



After adding all the above transforms, you should see the 10 steps on the right of the screen under **Transforms**. Click on **< Data flow** on the top left of the screen.



Data Export Options

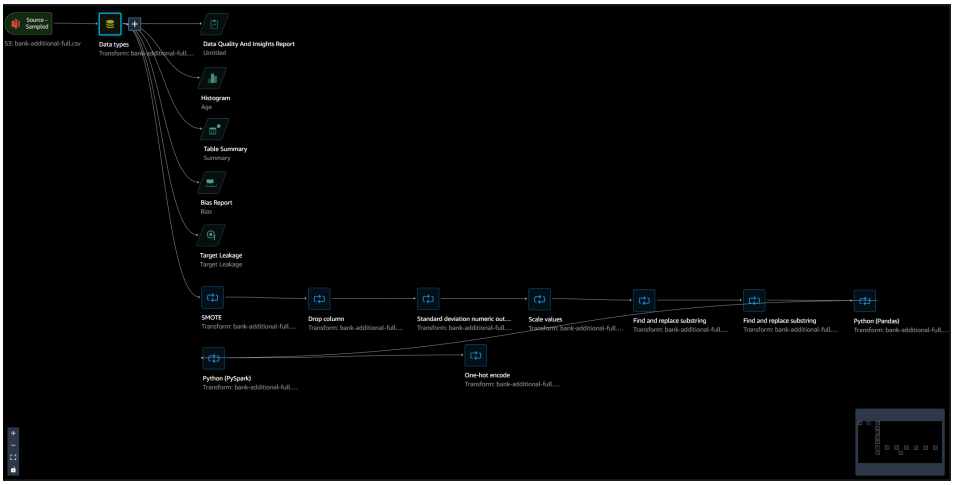
The ImmersionDay.flow file we created captures all of the transformations and analyses we have stitched together so far using the visual editor. This file persists every transformation step we have created and can be reused to modify/add new steps. After feature engineering we would like to persists the processed data in a storage that can be easily accessible or use the processed data for other down stream task including model training. We would look into a few of the export options and training capabilities integrated with Data Wrangler. To see a full list of the export options provided by Data Wrangler, please read this documentation.

Export to Amazon Feature Store | Export to Amazon S3 | Export to SageMaker Inference Pipeline

Amazon SageMaker Feature is a purpose-built repository where you can store and access features so it's much easier to name, organize, and reuse them across teams. SageMaker Feature

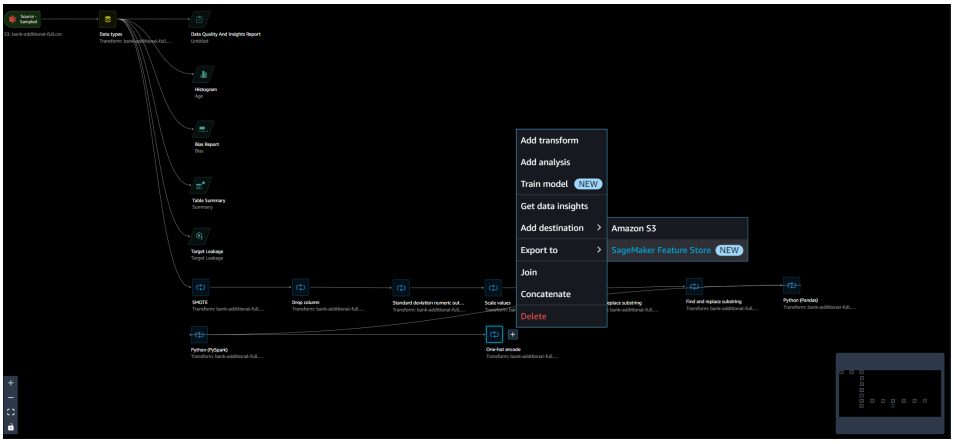
Store provides a unified store for features during training and real-time inference without the need to write additional code or create manual processes to keep features consistent. SageMaker Feature Store keeps track of the metadata of stored features (e.g. feature name or version number) so that you can query the features for the right attributes in batches or in real time using Amazon Athena, an interactive query service. SageMaker Feature Store also keeps features updated, because as new data is generated during inference, the single repository is updated so new features are always available for models to use during training and inference.

Data flow window should show all your transformation steps

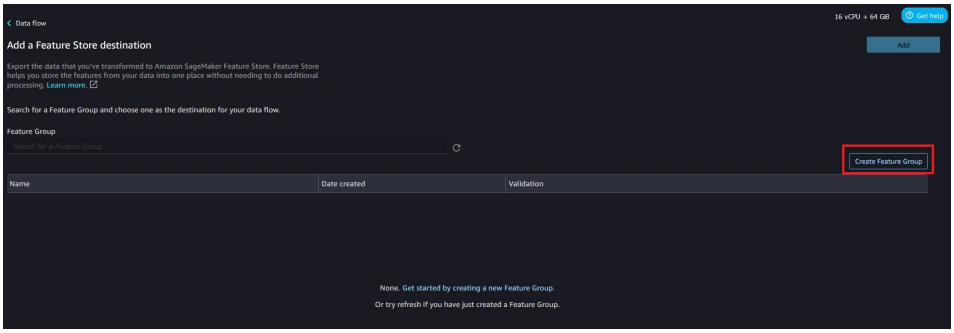


Destination nodes allow you to choose destination sinks where your transformed features must go to. The destination sink can either be Amazon Simple Storage Service (S3) or SageMaker Feature Store. Once you create the node, you can configure and kick-off a SageMaker Processing job which will run a distributed fully managed Spark job to scale your flow recipe on larger amounts of dataset with the same schema.

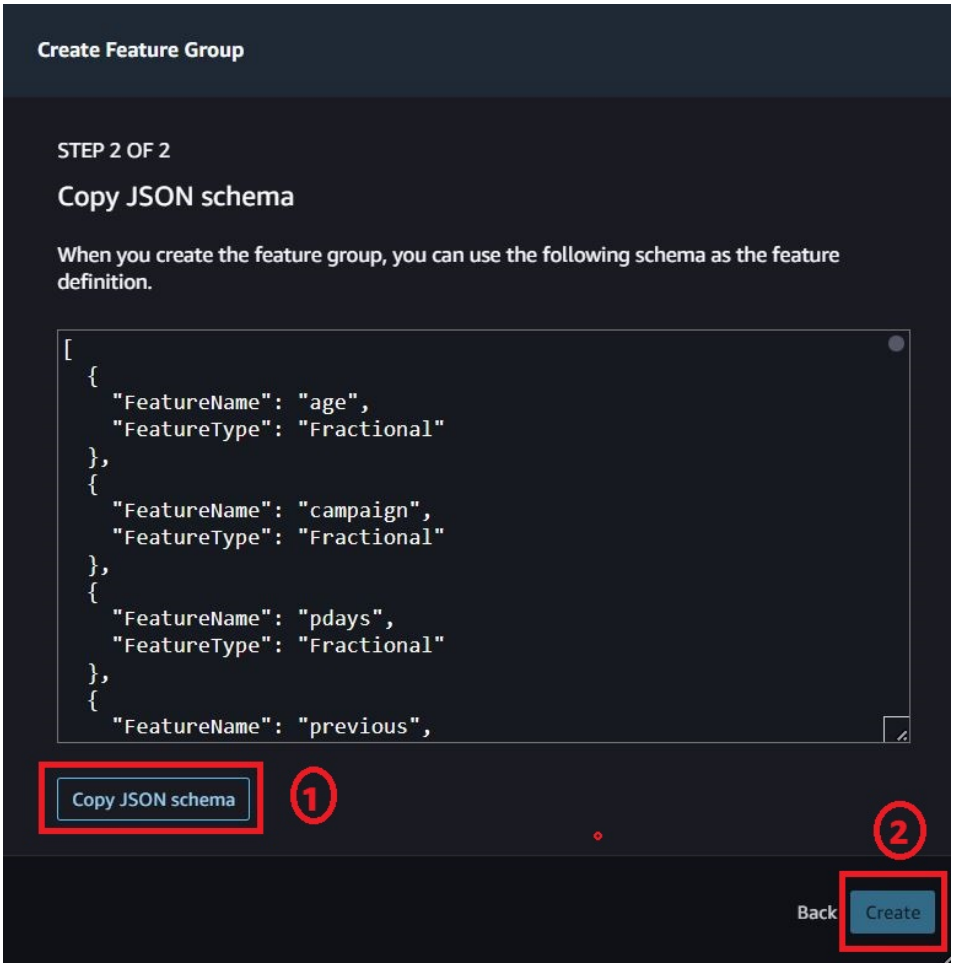
Let's export all the features that have been created in Data Wrangler to Feature Store. This can be done by choosing the + sign on the **One-hot encode** transform tile. From there, choose **Add destination→ Sagemaker Feature Store**



Choose **Create Feature Group** button. On the pop-up window, leave **Create "EventTime" column** box unchecked and choose **Next**.

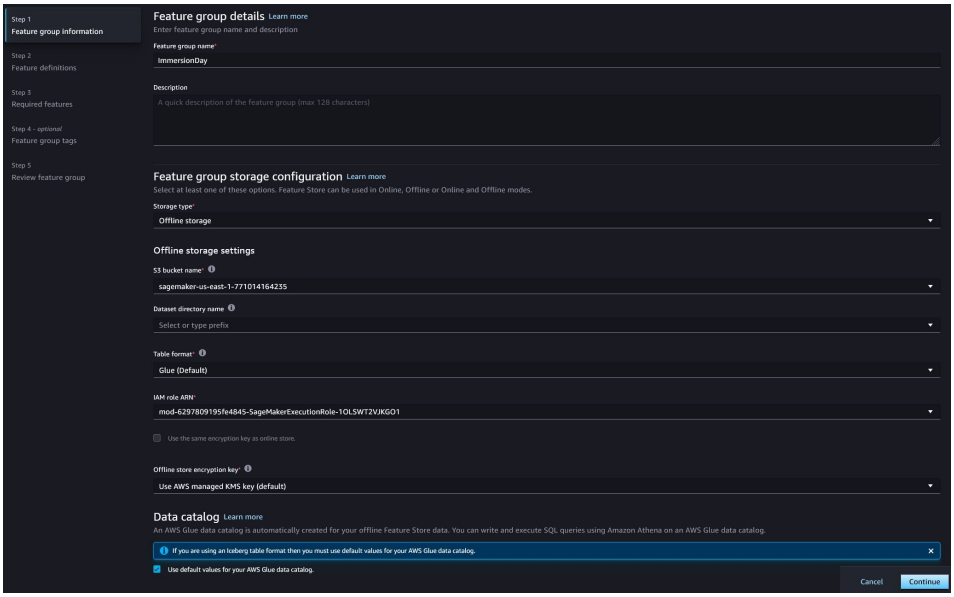


Choose **Copy JSON schema**, and choose **Create.** We would use the copied schema to create the Feature Store group.

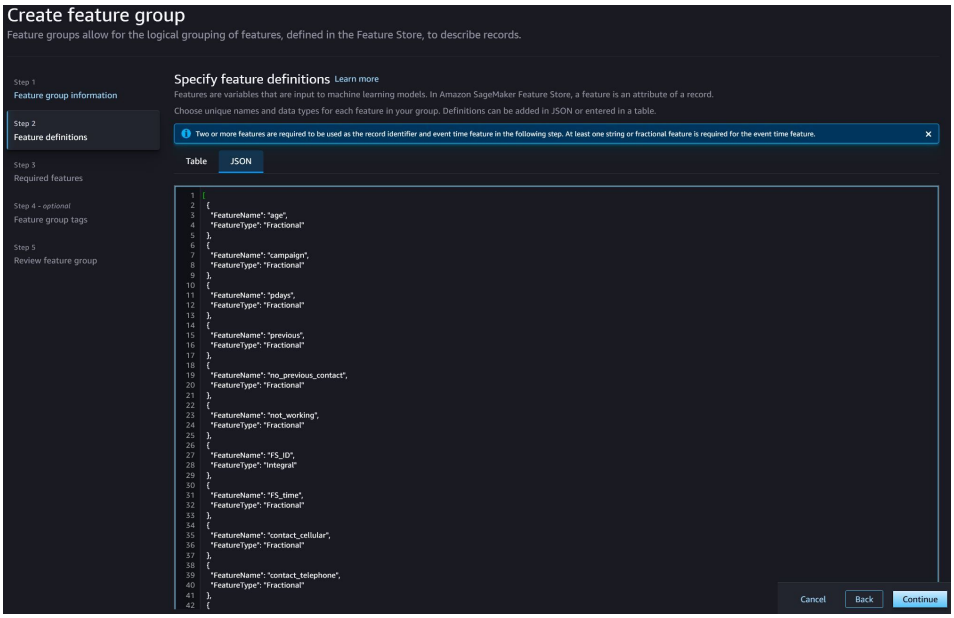


Enter `ImmersionDay` for **Feature group name**, choose **Offline storage** under **Storage type**. We don't need the online store as we will use Feature Store to retrieve features for training (not for real-time inference). Select `sagemaker-us-east-1-*` (may not be `us-east-1` depending on AWS region) under **S3 bucket name** and select `*-SageMakerExecutionRole-*` (type `sagemakerexecutionrole` to populate the result) under **IAM Role ARN.** Choose **Continue.**

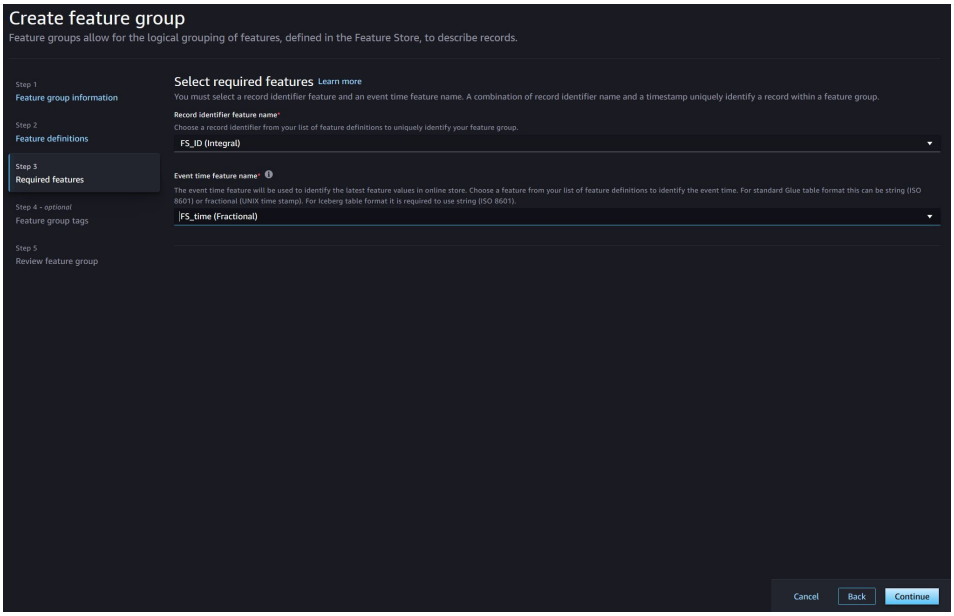
> Make sure you select the appropiate IAM role ARN to prevent any permission errors from creating the Feature Group.



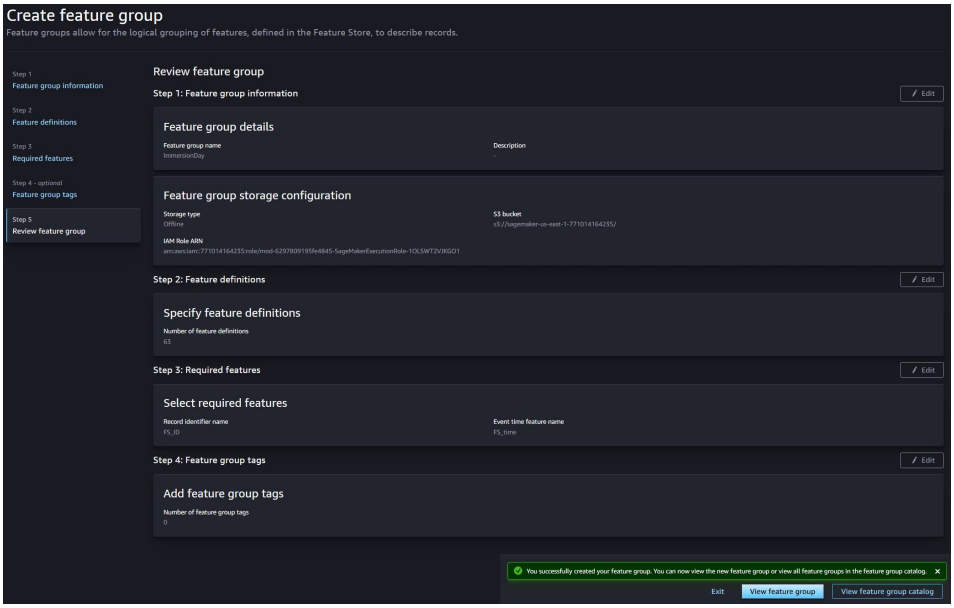
Choose **JSON** tab and paste the copied schema from the previous step. This is the schema of the transformed dataset from DataWrangler. The feature group we are creating needs to have the same data schema. Choose **Continue.**



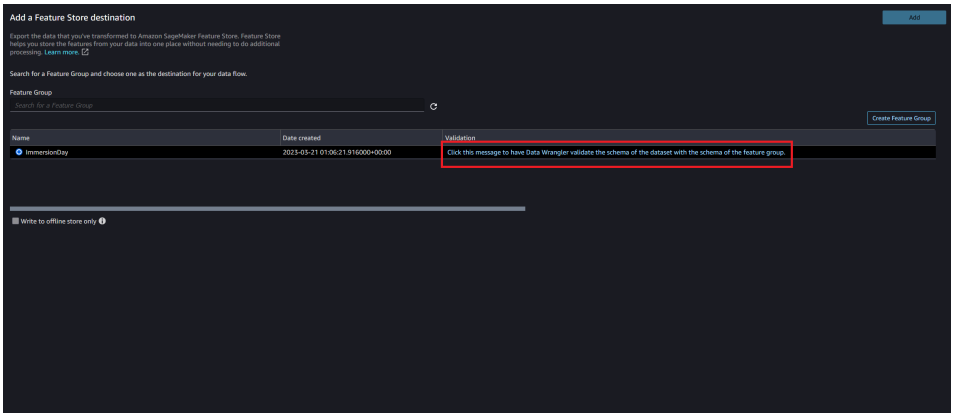
Select `FS_ID (Integral)` under **RECORD IDENTIFIER FEATURE NAME** and `FS_time (Fractional)` under **EVENT TIME FEATURE NAME,** choose **Continue.**



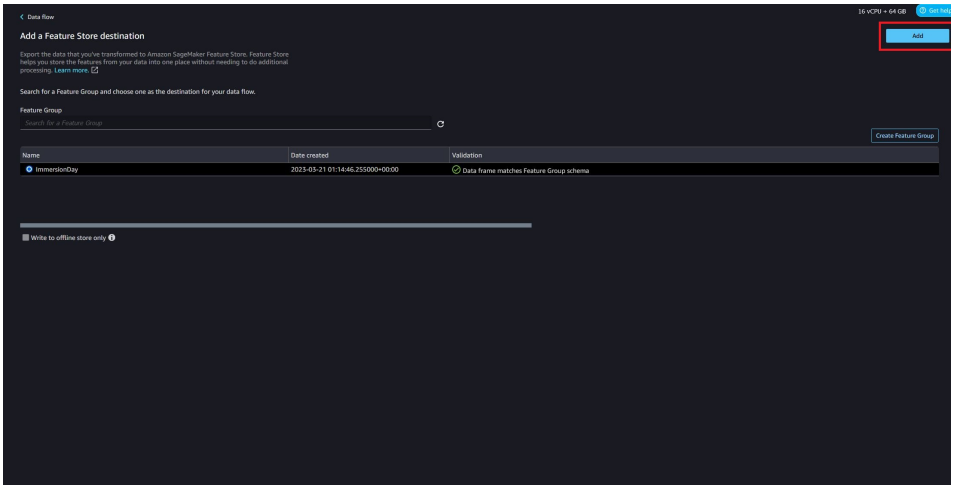
Choose **Continue** and choose **Create feature group**. You would get a feature group created success notification.



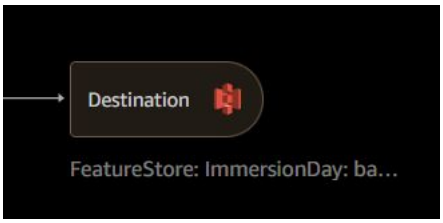
Go back to your DataWrangler flow, choose the plus sign on your last tile and add feature store as destination again. You should see the newly created feature group populate. Select the `ImmersionDay` feature group under **Name** tab and click on the message under the **Validation** tab to validate the schema of the dataset to that of the feature group.



Once validation is succesful, choose **Add.**



A feature store destination node will be added to your flow.

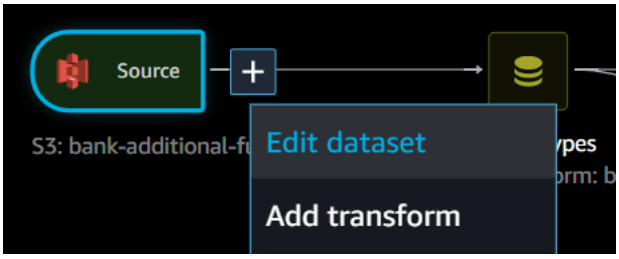


Before we move on to creating a SageMaker processing job, lets prepare the flow to accept a variable S3 file path, instead of the existing hardcoded one.

Parameterize the S3 data source path

When we imported the source dataset from S3, Data Wrangler registered the absolute path to that file. In this step, we'll parameterize that path, so we don't need to modify the import node/source data each time we wish to run the same transform flow against a different file.

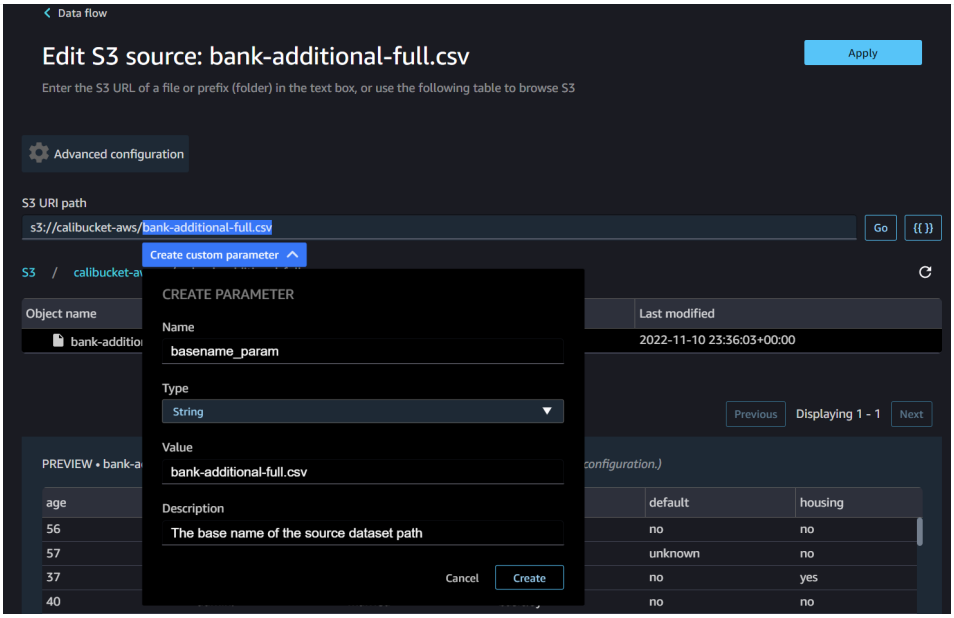
1. Click the **+** from the **Source** node, and click **Edit dataset**.



You will see the *S3 URI path* at the top, ending in the basename of the dataset file, in our case, `bank-additional-full.csv`.

2. Highlight just the *bank-additional-full.csv* portion of the path.

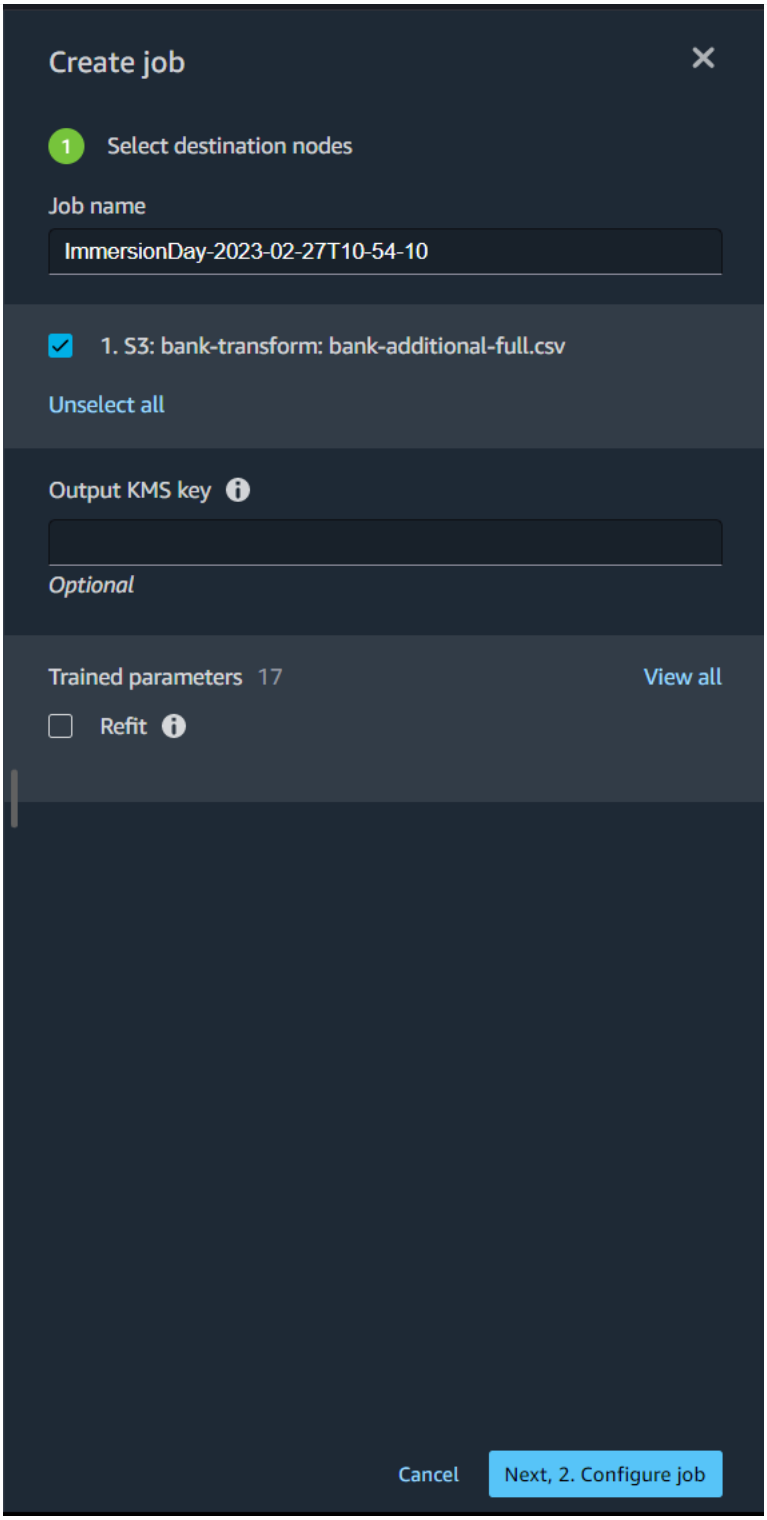
The **Create custom parameter** drop-down appears. Click it to open it up, and create the parameter as shown in the figure below.



3. Click **Create** to create the parameter.
4. Click **Apply** at the top of the screen to confirm the change.

Create Processing Job

Now that we've parameterized the filename, and defined a destination node, we can move on to creating the actual processing job to execute the flow. To do this, choose **Create job** on the top-right corner of your **Data flow** screen.

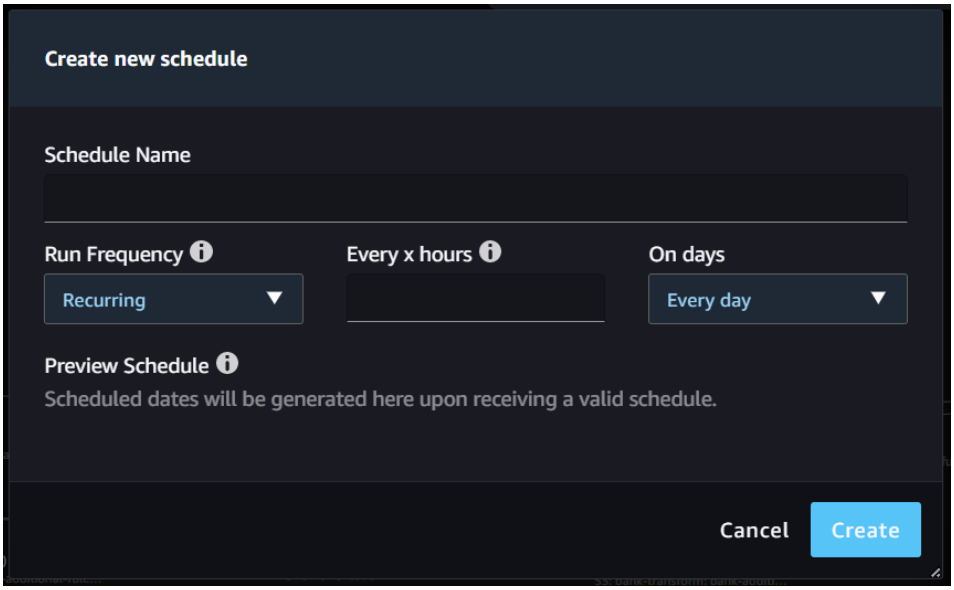


We will keep the **Output KMS key** empty as we will not be encrypting our data output for this example.

There are 17 trained parameters in our flow file. These trained parameters have been fitted on the available data in Data Wrangler with certain transforms including **One-hot encode**, **Scale values** and **Standard deviation numeric outliers**. These transforms depend on the available data and must be re-fitted when new data is available. Users can specify when they want to refit new transforms on their data by checking the **Refit** parameter when creating a Data Wrangler job. For a more in-depth walkthrough on the Refit feature see https://aws.amazon.com/blogs/machine-learning/refit-trained-parameters-on-large-datasets-using-amazon-sagemaker-data-wrangler/ We will keep the Refit paramter unchecked since we did not sample our dataset and imported the full dataset into Data Wrangler. Choose **Next, 2. Configure job.**

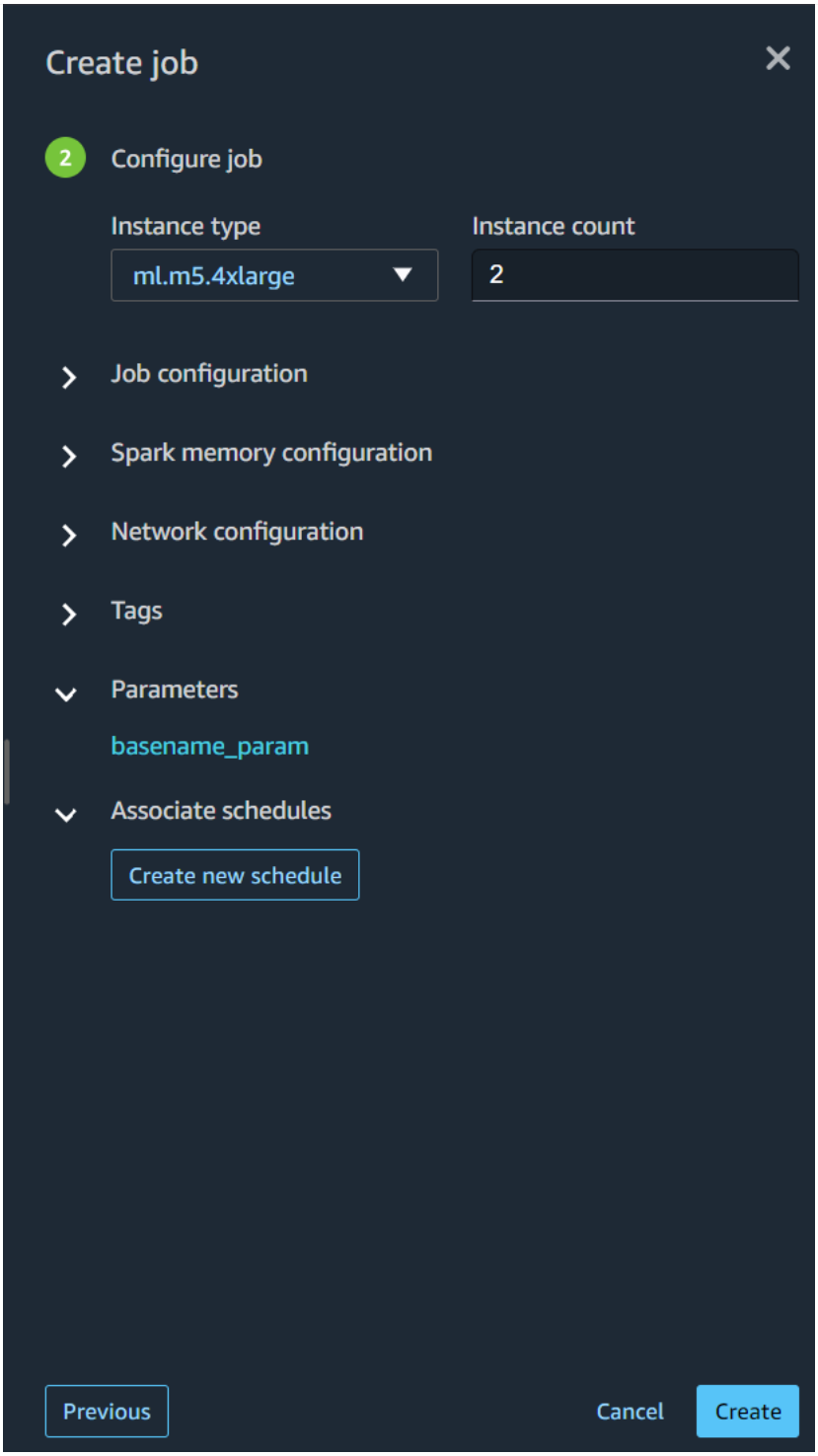
The *Configure job* panel presents you with additional configuarble parameters including **Instance type**, **instance count**, **Volume size**, etc. Some interesting features of Data Wrangler are:

1. **Associate Schedules**: You can schedule your Data Wrangler jobs to run at specific times and frequency. CRON expression provide additional flexibility in providing time schedules. To schedule Data Wrangler jobs, click on **Associate schedules** and **Create new schedule.** A pop-up appears where you can specify the Schedule Name and timing configurations. We will not be creating a schedule for this workshop.

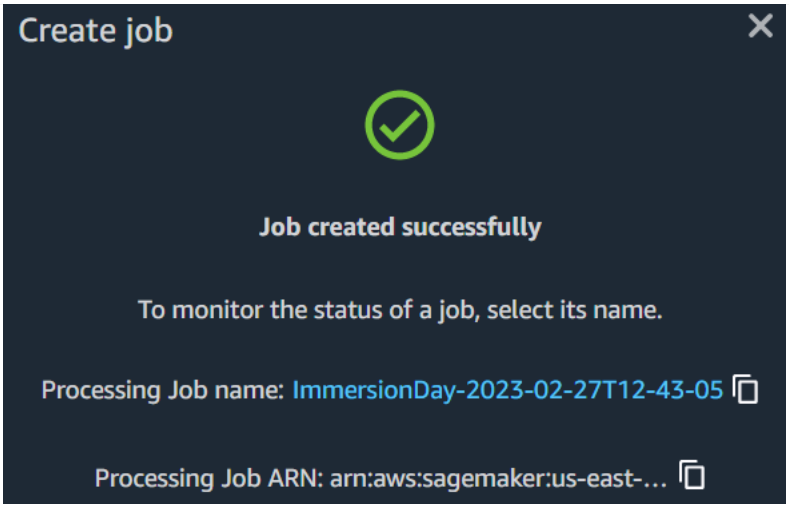


2. **Parameters**: We parametrized our source data path, `basename_param`, earlier which is available under the **Parameters tab.** Here you can change the value of `basename_param` to point to a different file name. But for this workshop we will leave it same.

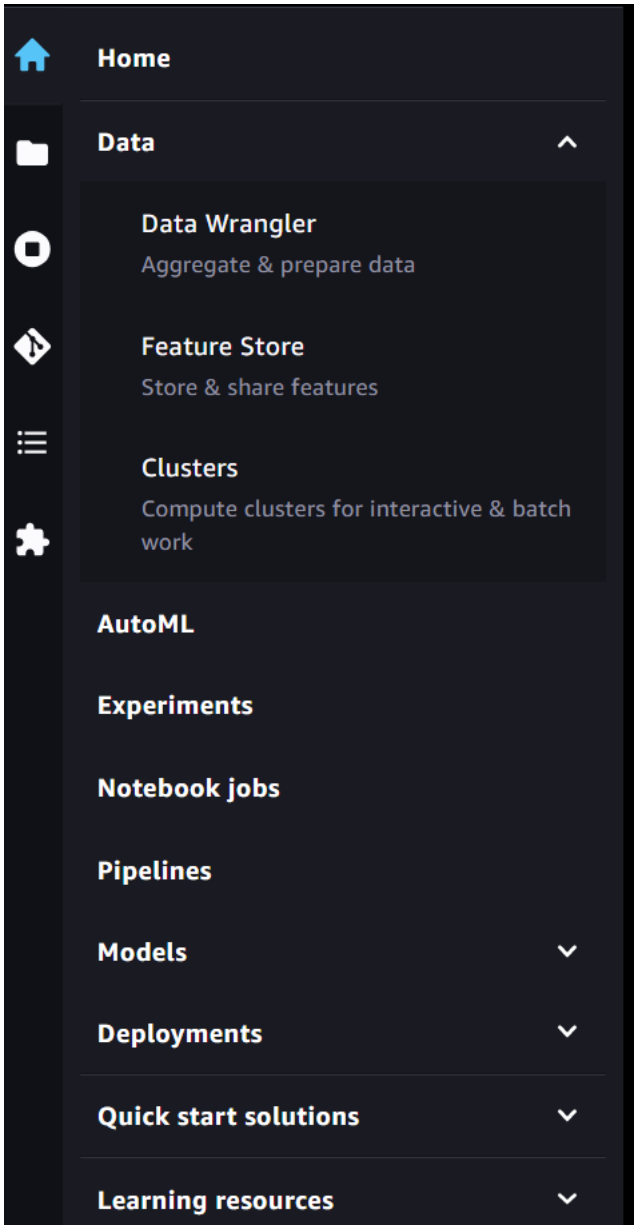
Click **Create** to create the processing job.



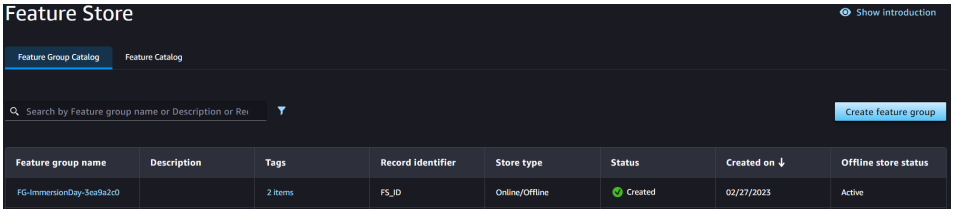
The confirmation screen will appear. We have created a Sagemaker processing job to execute the Data Wrangler flow we created, implement all the transforms we added and store the output in the feature group we created



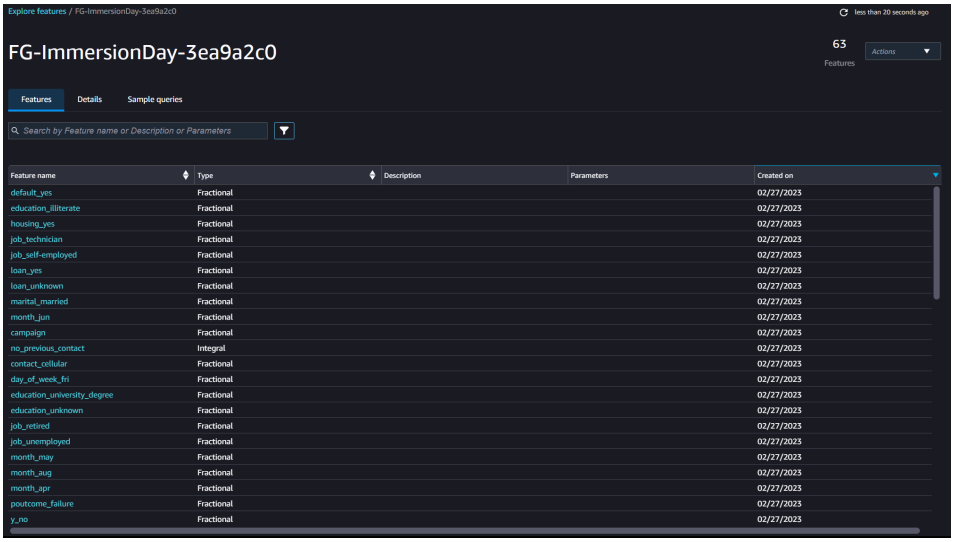
Now, let's view the newly created feature group. In the SageMaker Studio Notebook environment, choose the 'home' icon, choose **Data** and select **Feature Store**.



You will then be able to see the feature group created inside the Feature store:

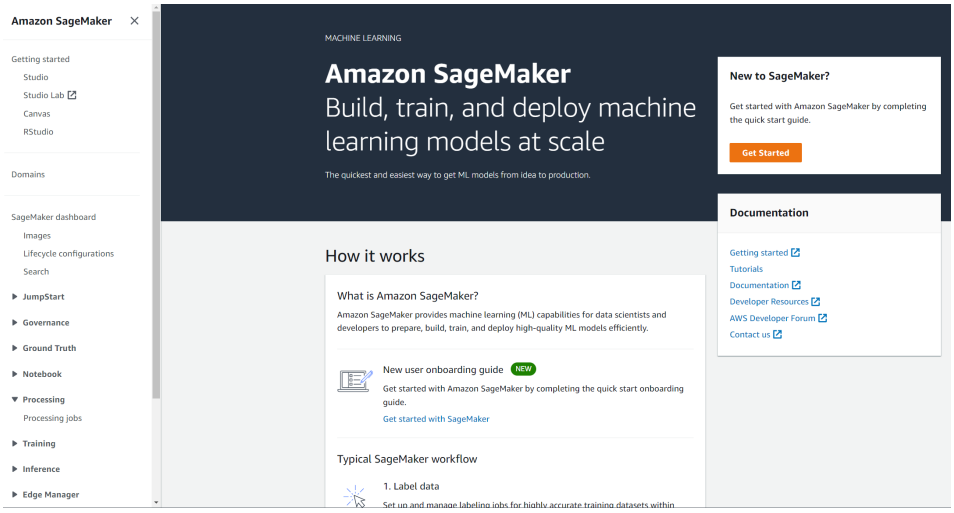


With the corresponding structure if you click on it:

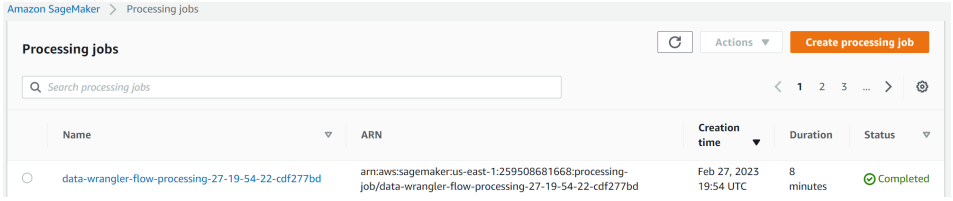


> You can copy the feature group name to a notepad. You will need it later in the lab.

You can go to the main Amazon SageMaker page in the AWS console and click on **Processing jobs**:



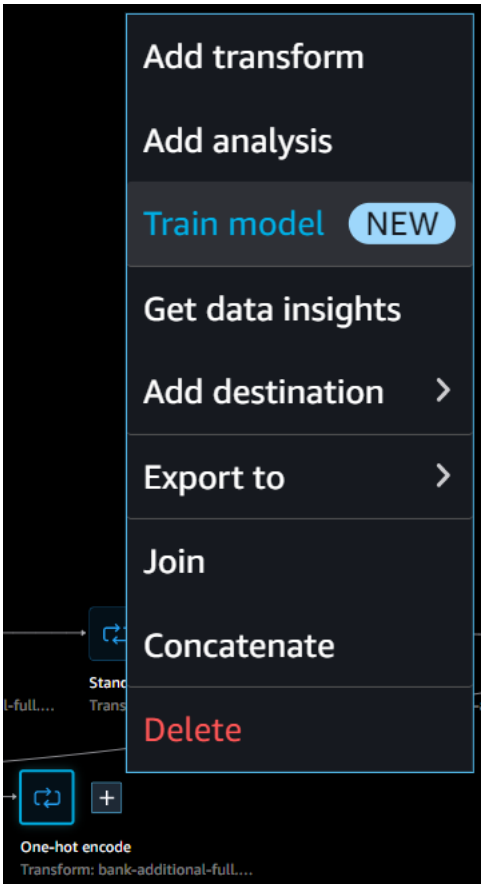
You will see a processing job for data wrangler ingestion into features store:



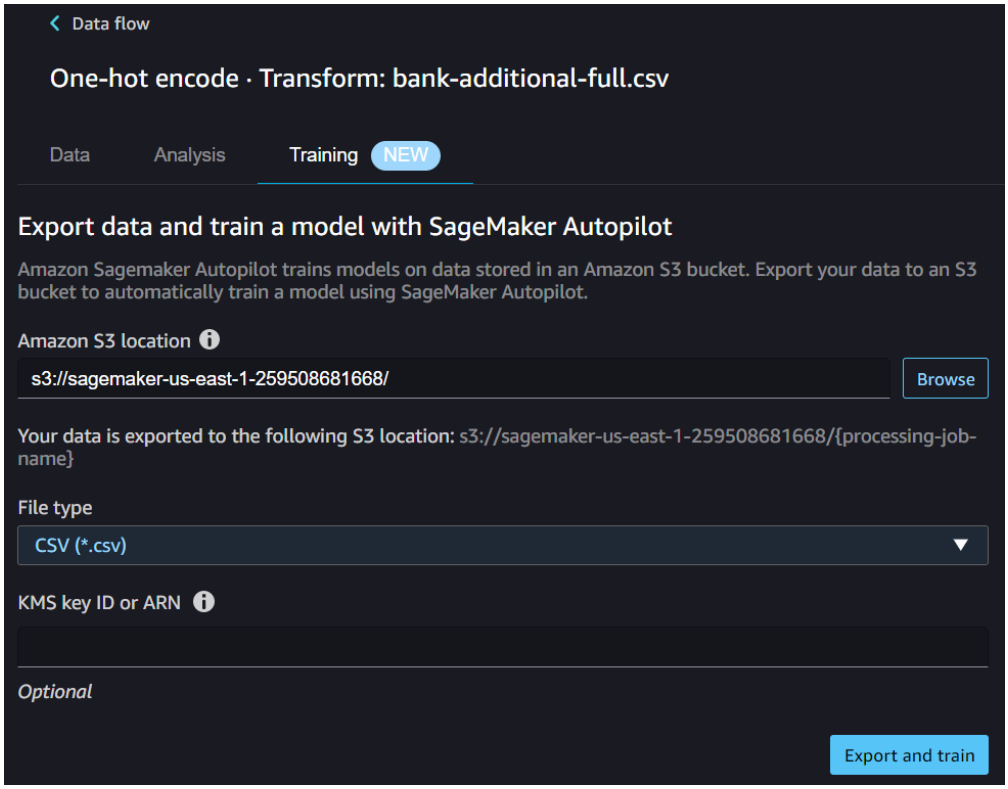
Click on the job name under **Name** to reach the **Job settings** menu which shows a complete information about the processing job.

Model Training with Amazon SageMaker Autopilot *(Optional)*

Data Wrangler provides a unified experience enabling you to prepare data and seamlessly train an ML model, all from within the tool. With just a few clicks, you can automatically build, train, and tune ML models using Autopilot, XGBoost, or your own algorithm, directly from the Data Wrangler user interface (UI). In this lab, we will focus on training with AutoPilot. To get started, choose the + sign on the **One-hot encode** transform tile. From there, choose **Train model**.



On the **Training** console, you can change the S3 output location and the output file type. However, for this workshop we will go with the default parameters.



Once export is successful, you are taken to the **Create an Autopilot experiment** page, with the **Input data** S3 location already filled in for you (as it was populated from the results of the previous screen.) More information on how to configure an Autopilot job is found in Lab 4 of this workshop.

Congartulations!!! You have succesfully processed and cleaned your data, also saved it to a persistent storage.

Conclusion

In this lab you have walked through the process of required environment setup and data engineering to clean and prepare your data for model building and training. In the next lab you will learn how to Train , Tune and Deploy an XGBoost model using SageMaker's Built-in XGBoost algorithm.