BDB 8.2 All DS Lab Quick Tutorial Workflows

Contents

BDB 8.2-Data Science Lab Workflow - 1

In Workflow 1, we'll explore the Data Science Lab, which allows us to effortlessly create projects and notebooks, upload external Python notebooks, build and register models, and establish a data pipeline to store the model's output in the data sandbox.

DS Lab is the ultimate platform for citizen data scientists, providing a low-code environment that empowers users to effortlessly organize their work and collaborate with others. Leveraging the advanced features of the BDB Platform, DS Lab takes data science to new heights, making it accessible and intuitive for both beginners and experts alike.

Workflow 1 is your gateway to a seamless data science experience. It's carefully designed to equip you with all the necessary tools and functionalities to work effectively on your data science projects. From project creation to model output, every step is streamlined to ensure your success.

Note - "When parking your vehicle, be sure to turn off both the engine and headlights."

To optimize resource consumption, it's essential to deactivate & unregister any active projects, pipelines, or models when you're finished with them. This step releases resources for other tasks and helps prevent potential issues.

Let's take care of this step first, then move on to the rest of your tasks.

For Sample Python notebook – Click here to download.

DS Lab Project Creation

Here, we will cover the step-by-step process of creating a new Data Science Lab Project and performing various Data Science tasks on the data.

To Begin,

- Click on the 'Create Project' tab to begin the process of creating a new project.
- Provide a suitable project name and a brief description that explains the purpose of your project. This will help you and others understand the project's context.

- Next, you need to select the algorithms that align with your specific use case. DS Lab offers various algorithm options, including Regression, Classification, Forecasting, NLP, and more. These algorithms are designed to cater to different types of machine learning tasks.
- DS Lab supports several Python frameworks, such as TensorFlow, PyTorch, PySpark, and Scikit-learn (which is the default). Depending on your project requirements and familiarity with these frameworks, choose the one that best suits your needs.
- Allocate appropriate resources to your project based on its size and complexity. DS Lab allows you to adjust resource allocation to ensure your project has enough computing power to run efficiently.
- Take advantage of the 'Idle Shutdown' feature. This feature allows you to specify a time limit for the project to remain idle. Once this idle time threshold is reached, the project will be automatically deactivated, freeing up resources for other tasks. This helps optimize resource utilization within DS Lab.
- You have the flexibility to incorporate external libraries into your project as needed. For example, if you require data from AWS S3, you can include the 'boto3' library to facilitate data retrieval.
- For more complex and large-sized projects that demand enhanced performance, you can configure a GPU (Graphics Processing Unit). Utilizing a GPU can significantly speed up computations and improve the overall performance of your project.
- Once you have made all the necessary configurations and settings, it's time to save your project. Congratulations! Your new project has been successfully created in DS Lab.
- From this point onwards, you can start working on your project, exploring data, training models, and analyzing results using the resources and tools provided within DS Lab. Enjoy your data science journey!

Create Notebook and Choose Dataset for Experimentation

Here, we will cover the step-by-step process of creating a new Notebook and upload and read data without manually writing code.

To Begin,

- Activate the project where you intend to work.
- Choose "Create Notebook" from the presented options after project activation.
- Wait for the kernel to start.
- Click on "Datasets" on the right-hand side to load the dataset.
- Add data from "Data Sandbox" by uploading a CSV file from your local machine. Here, the user can upload any CSV file for practice.
- Avoid using an existing file name and provide a sandbox name and description for the dataset.

- Click on the plus icon under "Datasets" to select the desired source and dataset for your experiments.
- Check the dataset checkbox to retrieve the code for writing the data into a dataframe, eliminating the need for writing lengthy code.
- Proceed with your experimentation based on your specific use case using the dataset.

Upload Notebook and Create Model

Here, we will cover the step-by-step process of Uploading an External Python Notebook and Creating a Classification Model on top of the PIMA Indian Diabetes Dataset.

For Dataset use below Git Hub Url:
"https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

To Begin,

- Click on "Upload Notebook" and select the desired file from your local machine. Wait for the kernel to start and the notebook to be successfully uploaded.
- Proceed with creating a classification model for the Pima Indians diabetes data.
- Import necessary libraries: Pandas, "model_selection," and "multinomialNB" from Scikit-learn.
- Read Pima Indians diabetes data from a CSV file using the Git URL, specifying column names, with the class column placed at the end as the target column.

```
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age',          'class']
```

- Define feature column (X) and target column (Y) within an array.
- Choose between the first or second method to define the array for X and Y variables.
- Define test size (0.33) and random seed value (7).
- Split data into training and test sets using "model_selection.train_test_split" to get X_train, X_test, Y_train, and Y_test.
- Apply the multinomial Naive Bayes classifier model to the training data.
- Save the model using the BDB DSLab library, specifying model name, model type = 'ml'.
- The X and Y variables, along with the estimator_type, are used to create a model explainer dashboard. We'll cover this in a separate workflow.
- Navigate to "Models" option, select your model, and the code cell will populate with the necessary code to load the model.
- Create a copy of the test data as a best practice.

- Use "nb.predict" function for model prediction, providing the model name and test dataframe as parameters.
- Execute the prediction and print the predicted data.
- Assess the model accuracy using "accuracy_score" function from Scikit-learn library.

Register Model

Here, we will cover the step-by-step process of Registering our Model to use it in data pipeline.

To Begin,

Within the notebook interface:

- Click on the "Models" tab located on the right side.
- Once in the "Models" section, select "View all models" to access the complete list of models.
- Within the list of models, locate the specific model that you want to register.
- After finding the model, you'll have the option to register it. Click on the "Register" button associated with the model.
- Congratulations! Your model has been successfully registered. It is now available for use in various data pipelines and experimentation scenarios.

Outside the notebook interface:

- Close the notebook to return to the main interface.
- In the main interface, click on the "Models" option to access the model management section.
- To find the model you just created, apply the "View all" or "Unregister" filter in the model management section.
- Once you've located the specific model, click on the "Register" button next to it.
- Congratulations! Your model has been successfully registered and is now ready for utilization in various data pipelines and experimentation scenarios.

Create a notebook for model input data script and export the script to the Data Pipeline

Here, we will cover the step-by-step process of writing a model input data script and exporting it to the Data Pipeline.

To Begin,

Step 1: Create the DataFrame for Model Input Data

- Name the notebook 'sklearn model input'.
- Develop a function to generate the input data and return it as a DataFrame.

- Import the pandas library.
- Read the CSV data from the git repository and use it to construct the DataFrame, making sure to define all the necessary column names.

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima- indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

Step 2: Create a Test DataFrame

- Create a test DataFrame by omitting the target column labeled as 'class' from the original DataFrame.

Step 3: Validate the Output

- Execute the function at least once to validate the output.
- Save your notebook to preserve any changes made.

Step 4: Export the Notebook Script to the Data Pipeline

- Select the 'Export' option in the notebook.
- Choose the desired cell to be included in the export.
- Click 'Next' and then 'Export'.
- The script can now be incorporated into your pipeline for further utilization.

Create Data Pipeline and Write data to Data Sandbox

Here, we will cover the step-by-step process of creating a data pipeline workflow to write data to the Data Sandbox.

To Begin,

- Open the data pipeline module.
- Create a new pipeline, providing the required name, description, and resources.
- From the Machine Learning tab, drag and drop the DSLab Runner component
- Provide a name and basic information (keep component as Real-time).
- In the Meta information section, choose "script runner" and provide other information like function type, project name, script name, and start function. Save the component.
- Create a new Kafka event, providing basic information like name and duration. Click on 'Add Event.'

- Drag and drop the event and link it with the DSLab Runner component.
- Drag and drop the DSLab Runner component again, this time choosing "model runner" in the Meta information.
- Select your project and model from the DSLab and link this component with the previous event.
- Create a new event and repeat the process to add the event. Drag and drop this event and connect it with the component.
- Look for 'Sandbox Writer' in the search option and drag and drop it.
- Provide basic information and, in the Meta information, specify the sandbox file name, file type, and save mode. Save the component and pipeline.
- Activate the data pipeline and access the live pipeline logs from 'Advanced Logs' and 'Logs Monitor.'
- Verify that the data is successfully written in the data sandbox.
- After successful data load, deactivate the pipeline.
- Go to the data sandbox in the data center to confirm the creation of a new data sandbox file.
- Perform data preparation and create a data store for visualization from the data sandbox.

Hope you will be able to create your Data Science Workflows. **Thank you!**

BDB 8.2: Data Science Lab Workflow – 2

Create forecasting models using inbuilt algorithms, utility and artifact features of DS Lab.

In Workflow 2, we'll explore the power and simplicity of creating forecasting models using inbuilt algorithms or boilerplate code feature of DS Lab. Additionally, we'll explore the utility and artifact features of DS Lab to ease and optimize code.

In this workflow, we'll cover the following key steps:

- DS Lab Project Creation.
- Upload Utility Script file.
- Create Forecasting Model using the N-BEATS algorithm and Save the Preprocessed Dataframe as an Artifact that we can use for further experimentation.
- Use Artifact data to create a forecasting model using the Random Forest Algorithm.

Note - "When parking your vehicle, be sure to turn off both the engine and headlights."

To optimize resource consumption, it's essential to deactivate & unregister any active projects, pipelines, or models when you're finished with them. This step releases resources for other tasks and helps prevent potential issues.

Let's take care of this step first, then move on to the rest of your tasks.

For Dataset and Utility File: Click Here to Download.

DS Lab Project Creation

Here, we will cover the step-by-step process of creating a new Data Science Lab Project and performing various Data Science tasks on the data.

To Begin,

- Create a new project, name it, and provide a brief description.
- Choose the "Forecasting" algorithm.
- Specify required environment, resources, and idle shutdown time.
- Avoid excessive resource allocation.
- Save the project.

Upload Utility Script file

Here, we will cover the step-by-step process of uploading the utility script file, which includes a standard preprocessing script.

To Begin,

- Activate and open the project.
- Navigate to the utility tab in the project interface.
- Enter the utility name and provide a description for the utility.
- Select a utility script from your local machine by browsing or uploading it to the platform.
- Click on the "Save" button to store the utility in the project.
- If you need to make changes or updates to the utility script, click on the "Edit" button.
- Update the script as needed, and then save the changes.

Create Forecasting Model and Artifact

Here, we will cover the step-by-step process of creating a forecasting model using inbuild Algorithm feature and store preprocessed dataframe in Artifact.

To Begin,

- Create a new notebook and give it a name for the script.
- Load the data and create a dataframe by clicking on "Dataset." Choose the dataset if already uploaded, or upload your data if not done yet.
- Refer to DS Lab workflow 1 to learn how to load data in the "Dataset" tab. The code will be automatically populated by clicking the checkbox, saving you the effort of writing the code. Modify it as per your requirements.
- Preprocess the raw data using a utility script. Import the utility script by typing the name of the .py or python file and provide an alias.
- Use the preprocess_data function of the script on the raw data to create the processed dataframe "df."
- For modeling, select the inbuilt algorithm option by clicking on "Algorithm," then choose "Forecasting," and select the N-BEATS model.
- The complete code for the model will be automatically populated by clicking the checkbox. Provide data fields like the full dataframe, date column, and target column (e.g., "sale").
- Execute the cell to start predicting. Wait until the prediction process is completed successfully.
- The results will be plotted using the Matplotlib library, showing the actual and predicted chart.
- Use "modelname.predict" to predict results for different scenarios.
- Explore the Artifact features of DS lab and their benefits.
- Save the preprocessed dataframe as an artifact by selecting the "Save Artifact" option from the right-hand side.

- Provide a name for the dataframe and a file name with the .csv extension to store the artifact successfully.

Create Forecasting Model using Artifact data as input

Here, we will cover the step-by-step process of creating a forecasting model using the inbuild Algorithm feature and previously saved Artifacts as input data.

To Begin,

- Create another notebook.
- Easily access the artifact data.
- Provide a name for the notebook.
- Click on "Dataset" and choose the "Sandbox" option to find the artifact file.
- Add and choose the artifact file to create a dataframe.
- Select the "Random Forest" model from the available algorithms for forecasting.
- Specify data information, including target and date columns.
- Run the cell to generate the forecasting successfully.
- Save your notebook and the deactivate DS Lab project once done.

Hope you will be able to create your Data Science Workflows. **Thank you!**

BDB 8.2: Data Science Lab Workflow – 3

In Workflow 3, we harness the power of DSLab's Notebook Model Explain ability for churn analysis. Firstly, we create a robust DS Lab model within the Tensor environment. Next, we seamlessly export this model into the Churn Pipeline, enhancing its predictive capabilities. The transformed data is then written into the data sandbox, ensuring a secure and efficient data storage process. Finally, we craft a compelling business story, leveraging the insights gained from the explainable model, to drive actionable strategies and achieve optimal customer retention and growth. Experience the synergy of advanced technologies in this innovative workflow, paving the way for informed and data-driven decisions.

Note - "When parking your vehicle, be sure to turn off both the engine and headlights."

To optimize resource consumption, it's essential to deactivate & unregister any active projects, pipelines, or models when you're finished with them. This step releases resources for other tasks and helps prevent potential issues.

Let's take care of this step first, then move on to the rest of your tasks.

For sample Utility Files: Click Here to Download.

Let's get started and understand how to work with the DS Lab Notebook. In this Workflow, we will Upload Churn prediction notebook, load and explore data, perform some data analysis, and visualize the results. Let's begin

After accessing the home screen of the platform, select the "DS Lab" Plugin from the app menu. This will take you to the home page of the Data Science Lab.

On the home page, you'll find a list of projects. Each project contains essential information like its name, description, environment, resource allocation type, libraries, and action items like version control, sharing, editing, and more.

"Here's the 'Churn Prediction' project, already set up with a tensor environment." This project is already active if it’s not please activate it First.

- Click on the 'Churn Prediction' project to see the existing notebooks."
- On the right-hand side, you'll find two buttons: 'Upload Notebook' and 'Create Notebook.'"
- Click on the 'Upload Notebook' button to upload a notebook to the project.
- Great! You are now on the notebook page where you can upload your existing notebook.
- Simply select the notebook file you want to upload and follow the instructions."

Once the upload is complete, your notebook will appear on the notebook page. Congratulations! You've successfully uploaded your notebook to the 'Churn Prediction' project. Now, you can access, edit, and run your notebook in the DS Lab environment. Happy data exploration and modeling!

```
from Notebook.DSNotebook.NotebookExecutor import NotebookExecutor

nb = NotebookExecutor()

data = nb.get_data('59801689926743119', '@SYS.USERID', 'True', {}, [])

data['Churn'] = data['Churn'].map({

   'No' : 0,

   'Yes' : 1

})
```

data.head(3)

"First, we import the necessary class and create an instance of it. Then, we call the get_data method to fetch the dataset.Now, let's process the data to prepare it for analysis. We use the map function to convert 'No' to 0 and 'Yes' to 1 in the 'Churn' column." Here are the first three rows of the processed dataset

data.dtypes

let's gain insights into the data types of each column in our dataset using the data.dtypes attribute.

The table above showcases the data types associated with each column. These data type details provide essential information for further data processing and analysis.

data.select_dtypes('object').columns

Let's take it a step further and explore how to select specific columns with data type 'object' from our dataset. The table above shows the columns that have been selected based on their data type, which is 'object'. This technique allows us to narrow our analysis to the relevant columns for in-depth exploration.

data.select_dtypes('number').columns

Now, let's explore how to select columns with numerical data types from our dataset.The table above shows the columns that have been selected based on their numerical data types. This approach allows us to concentrate on numeric data for advanced quantitative analysis.

```
import numpy as np

from sklearn.preprocessing import OneHotEncoder, MinMaxScaler


def preprocess_categorical(df_in, categorical_columns):

    df = df_in[categorical_columns].copy()

    ohe = OneHotEncoder()

    df_cat = ohe.fit_transform(df);

    df_cat = df_cat.todense()

    return df_cat, ohe
```

```
def preprocess_numerical(df_in, numerical_columns):
    df = df_in[numerical_columns].copy()
    scaler = MinMaxScaler()
    df_num = scaler.fit_transform(df);
    return df_num, scaler

def preprocess_data(df_in, categorical_columns, numerical_columns):
    df_cat, ohe = preprocess_categorical(df_in, categorical_columns)
    df_num, scaler = preprocess_numerical(df_in, numerical_columns)
    X = np.concatenate((df_cat, df_num), axis=1)
    n_cat_out = df_cat.shape[1]
    n_num_out = df_num.shape[1]
    return X, ohe, scaler, n_cat_out, n_num_out

def invert_preprocessing(df_in, ohe, scaler, n_cat_out, n_num_out):
    n_cat, n_num = ohe.n_features_in_, scaler.n_features_in_
    cat_inv = ohe.inverse_transform(df_in[:, :n_cat_out])
    num_inv = scaler.inverse_transform(df_in[:, -n_num_out:])

    cat_inv = pd.DataFrame(cat_inv, columns=ohe.feature_names_in_)
    num_inv = pd.DataFrame(num_inv, columns=scaler.feature_names_in_)
    df_out = pd.concat((cat_inv, num_inv), axis=1)

    return df_out




categorical_columns = ['gender', 'Partner', 'Dependents', 'PhoneService',

        'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',

        'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',

        'Contract', 'PaperlessBilling', 'PaymentMethod']


numerical_columns = ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']


X, ohe, scaler, n_cat_out, n_num_out = preprocess_data(data, categorical_columns,
numerical_columns)

y = data['Churn']
```

the code aims to preprocess data for machine learning tasks by transforming categorical variables into a binary representation (one-hot encoding) and scaling numerical variables to a fixed range (min-max scaling). However, it is important to ensure the DataFrame data is correctly defined with the specified columns for the code to function properly.

invert_preprocessing(X, ohe, scaler, n_cat_out, n_num_out)

By calling invert_preprocessing with the appropriate inputs, one can obtain the original DataFrame with the categorical and numerical columns in their original format before any one-hot encoding or min-max scaling was applied. This function is useful for reverting the processed data back to its original state, especially when it is necessary to interpret or analyze the data in its original form after applying machine learning algorithms that required preprocessing. However, it's important to ensure that the original DataFrame data used in the preprocessing matches the structure and order of columns used during the preprocessing steps for accurate inversion.

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

model = RandomForestClassifier()

model.fit(X_train, y_train);

preds_train = model.predict(X_train)

preds_test = model.predict(X_test)

print(classification_report(y_train, preds_train))

print(classification_report(y_test, preds_test))

the code performs data splitting, model training, prediction, and evaluation using the random forest classifier. The classification reports provide a detailed analysis of the model's performance on both the training and testing sets, giving insight into its accuracy, precision, recall, and F1-score for each class in the target variable y.

```
# make column transformer object to use for data preprocessing in pipeline

from sklearn.compose import ColumnTransformer

categorical_columns = ['gender', 'Partner', 'Dependents', 'PhoneService',

    'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',

    'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',

    'Contract', 'PaperlessBilling', 'PaymentMethod']


numerical_columns = ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges']

ct = ColumnTransformer(

  [

     ('ohe', OneHotEncoder(), categorical_columns),
```

('scaler', MinMaxScaler(), numerical_columns)

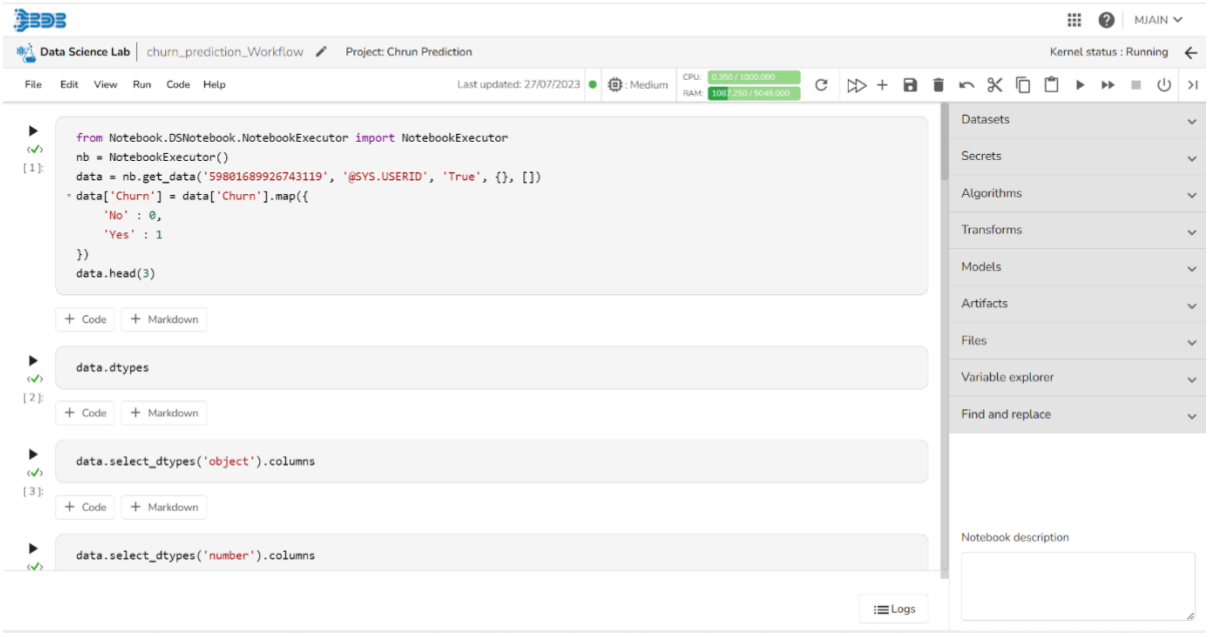
],

remainder='drop'

)

X_trans = ct.fit_transform(data);

By using this ColumnTransformer object ct, you can easily integrate it into a machine learning pipeline for efficient preprocessing of both categorical and numerical features. The pipeline would allow you to apply this preprocessing consistently on both the training and testing data without data leakage, making it easier to train and evaluate machine learning models effectively.



```
[1]:
from Notebook.DSNotebook.NotebookExecutor import NotebookExecutor
nb = NotebookExecutor()
data = nb.get_data('59801689926743119', '@SYS.USERID', 'True', {}, [])
data['Churn'] = data['Churn'].map({
    'No' : 0,
    'Yes' : 1
})
data.head(3)
```

```
[2]:
data.dtypes
```

```
[3]:
data.select_dtypes('object').columns
```

```
data.select_dtypes('number').columns
```

<u>Enocding part</u>

import pandas as pd

from sklearn.preprocessing import LabelEncoder

# Instantiate the LabelEncoder

label_encoder = LabelEncoder()

# Fit and transform the series to label encode the values

encoded_data = label_encoder.fit_transform(y_train);

# Create a new Series with the encoded values

encoded_series = pd.Series(encoded_data);

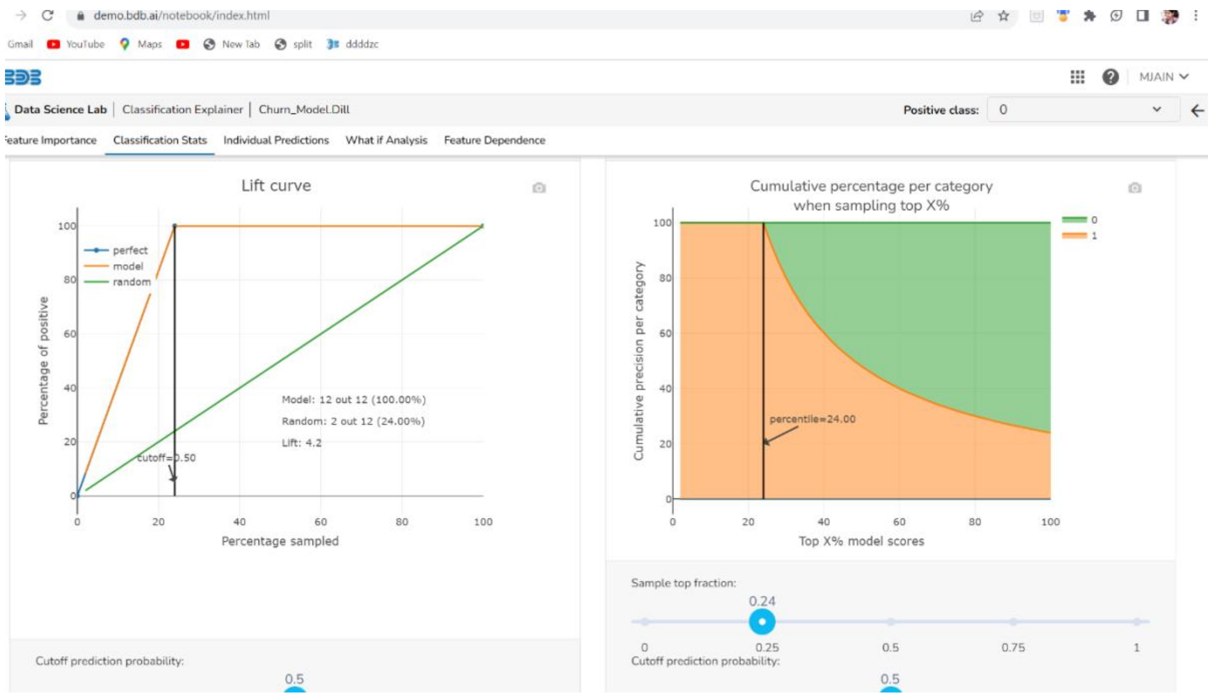
encoded_series

It imports the required libraries, pandas for data manipulation and LabelEncoder from sklearn.preprocessing for label encoding. The LabelEncoder is instantiated, creating an instance of the label encoder. The fit_transform() method is used to both fit the label encoder to the y_train series and transform it to label encode the categorical values. The result is stored in the encoded_data variable as a NumPy array. Finally, a new pandas Series called encoded_series is created using pd.Series() to store the label-encoded values in a more manageable and structured format for further analysis or usage.

let's save the Data Transformation Model. To do this, click on the "More Actions" three dots and select "Save Model" from the dropdown.

A code snippet will be populated. You'll need to specify the model as 'ct', and let's name it 'churn_preprocess'. The model type should be 'DP' for data transformation pickle files.

Now, let's create the Machine Learning Model. Similar to before, click on the "More Actions" three dots and select "Save Model". This time, specify the model name as 'churn_model' and set the model type as 'ml'.. This model was created using the 'X_train' and 'encoded_series' datasets and is intended for classification tasks.



Great! Let's ensure the models are saved correctly. Move to the right-hand side and check the "Transform" and "Model" tabs.

- You should see the saved models listed here - 'churn_preprocess' under the 'Transform' tab and 'churn_model' under the 'Model' tab.
- Now, we'll register the Data Transformation Model. Click on the "Transform" tab and select "All" from the filter to see all the saved models.
- Look for the 'churn_preprocess' model, then select the checkbox. Click on the three dots and choose the "Register" option.
- Next, let's register the Machine Learning Model. Click on the "Model" tab and follow the same steps as before.
- Find the 'churn_model' model, select the checkbox, click on the three dots, and choose the "Register" option.
- And there you have it! You've successfully saved and registered your Data Transformation and Machine Learning models

Now, you can use these models in your pipelines and experiments with ease

Model Explainer:

let's go ahead and Understand how you can perform model explanation and feature importance analysis for your Machine Learning model. For this demonstration, let's assume you have already trained a machine learning model and have access to the test dataset on which you want to explain the model's predictions.

- Just go to model section and clicks on three dots and select explainer option for the selected Model then you navigate to the Model explainer page
- Within the model explanations, feature importance analysis is a crucial aspect. This analysis helps determine which features had the biggest impact on the model's predictions. By examining feature importance, you gain insights into the key factors influencing the model's decision-making process.
- 'Classification Stats' tab provides various statistics regarding the classification model. Here, you can access a range of performance metrics that evaluate the model's accuracy and effectiveness. These metrics offer insights into how well the model performs in classifying positive and negative instances.
- Individual Prediction: The individual prediction section displays the predicted probability for each target label. It provides insights into how the model assigns probabilities to different classes for a specific observation.
- What If Analysis: The What If Analysis allows stakeholders to understand the potential consequences of different scenarios or decisions. In this analysis, you can change the values

of selected variables to see how the outcome would change. It helps identify the sensitivity of the outcome to different inputs and which variables are most important.

- Feature Dependence
- The feature dependence analysis explores the relationship between feature values and their impact on predictions. This analysis allows you to investigate how the model uses features in line with intuitions or learn about the relationships the model has learned between input features and predicted outcomes

Upload Util Notebook

- Clicks on back icon
- you will navigate to the notebook page
- just clicks on upload button and upload your notebook for enrichment the data
- let me upload enrichment notebook
- go back to notebook section and clicks on upload button and upload churn_pred util notebook
- once it's uploaded you can see your notebook

  Step 1: Importing Libraries and Setting up Logging

  Step 2: Global Variables and Connection Creation

  Step 3: Function for Executing ClickHouse SELECT Queries

  Step 4: Caching Prediction Data

  Step 5: The Main Function - func

```
from clickhouse_driver import Client

import pandas as pd

import logging

logging.basicConfig()

logger = logging.getLogger(__name__)

logger.setLevel(logging.INFO)


data_counter = 0
```

```
pred_cache = pd.DataFrame()


def create_client(host, port, database, user, password, use_numpy=False):
    global client
    if 'client' in globals():
        # if client already exists, just return it
        return client
    logger.info('Creating new connection')
    client = Client(host, port, database, user, password, settings={'use_numpy':use_numpy})
    return client


def ch_select(query, client):
    # execute a select statement. return data as a dataframe
    data = client.execute(query, with_column_types=True)
    df = pd.DataFrame(data[0], columns=[i[0] for i in data[1]])
    # type_dict = {k:map_dtype(v) for k,v in data[1]}
    # df = df.astype(type_dict)
    return df


def cache_pred_data_batch(df):
    global pred_cache
    pred_cache = pd.concat((pred_cache, df), axis=0)
```

```
def func(df_in, orig_table_name, host, port, database, user, password):
   global data_counter, client, pred_cache, original_table_len


   client = create_client(host, port, database, user, password)


   data_counter += df_in.shape[0]
   logger.info('==================================================')
   logger.info(f'Current data counter: {data_counter}')


   if 'original_table_len' not in globals():
      original_table_len = ch_select(f'select count(*) from {orig_table_name}', client).iloc[0,0]


   logger.info(f'Original table length: {original_table_len}')


   if data_counter == original_table_len:
      cache_pred_data_batch(df_in)


      orig_data = ch_select(f'select * from {orig_table_name}', client).reset_index(drop=True)
      pred_data = pred_cache.copy().reset_index(drop=True)


      logger.info('Concatenating predictions to original data')
      df_out = pd.concat((orig_data, pred_data[['predictions']]), axis=1)


      return df_out
```

```
else:
    cache_pred_data_batch(df_in)
    logger.info(f'Finished processing. Size of pred cache: {pred_cache.shape[0]}')
```

The func function is the heart of this script. It takes several parameters, including the input DataFrame (df_in), the name of the original table (orig_table_name), and the ClickHouse connection details (host, port, database, user, password). Upon receiving data, the function increments the data_counter by the number of rows in the input DataFrame. It also logs the current data counter, allowing us to track the progress of data processing. If the total number of processed rows (data_counter) equals the original table length, it means all the data has been processed. The script then performs the following steps:

- Caches the prediction data batch using cache_pred_data_batch.
- Retrieves the original data from ClickHouse and the cached prediction data, both as pandas DataFrames (orig_data and pred_data, respectively).
- Concatenates the prediction data with the original data based on the 'predictions' column and returns the resulting DataFrame as df_out.
- If data processing is not yet complete, the script caches the prediction data and logs the size of the prediction cache.
- go back and export script into pipeline
- clicks on export icon and select the first checkbox and clicks on export then your script will be exported to the pipeline

Finally, after exporting the script into a pipeline, it seems like the pipeline will likely utilize the exported script to execute the data enrichment and churn prediction tasks in a more automated manner.

Pipeline Workflow

Let’s move to pipeline flow.

- First, locate and select the Data Pipeline Plugin from the app menu. This will take you to the pipeline home page.
- Next, look for the create icon and click on it. You'll see the option to create a new pipeline. Click on the '+' icon to proceed.
- Great! Now, let's specify the details for our end to end DS Lab pipeline. Enter a suitable name for your pipeline, such as Churn_Prediction_Workfow3. For the description, briefly describe the workflow, such as End to End DS Lab Workflow '

- Now, choose the resource allocation type based on your requirements. This feature allows you to deploy the pipeline with high, medium, or low-end configurations, depending on the velocity and volume of data that the pipeline must handle."
- Once you're done with the configuration, click on the save button to save your pipeline."

Congratulations! You've successfully created the pipeline.

Once the pipeline is saved in the pipeline list, you can add components to the canvas to create your pipeline workflow or dataflow.

To add a component, simply drag the required component from the Component Palette, located on the left side of the user interface, and drop it onto the canvas. You can configure each component to define your pipeline workflow. The Pipeline Editor displays the Component Palette, which contains various components like Reader, Writer, Transformation, Consumer, Producer, Machine Learning, and more. Use these components to design your pipeline according to your specific requirements

- Drag and drop the Sandbox Reader component onto the canvas
- Now, select 'realtime' as the invocation type and move to the Meta Information tab."
- In the Meta Information tab, choose 'network' as the storage type and 'csv' as the file type."
- Next, select the desired Sandbox name from the dropdown. This should be a sandbox you previously uploaded your data to
- Now, choose the Sandbox file you want to process from the Sandbox file dropdown.
- Make sure to check the 'header' and 'infer schema' checkboxes to handle the data's header and automatically infer the schema
- Once you've configured the component, click on the save button to save your settings."

Congratulations! You've successfully configured the Sandbox Reader component for realtime data processing

- Now, let's create an event and connect it to the component."
- Click on the "Event Panel" icon located in the toolbar. This action will open the event panel, which allows you to manage events.
- Inside the event panel, locate and click on the "Add Event" button. This will initiate the process of adding a new event.
- In the event creation Page, configure all the mandatory fields. Make sure to fill in all the required information accurately.
- Set the partition for the event. The partition is a setting that specifies how the event should be categorized or grouped. In this case, set the partition value to 1.
- Once all the information is filled and the partition is set, click on the "Save" button to create the event. The event will now be added to the system.

- To display the event on the canvas, locate the event component in the event panel. You can usually find it by its icon or name.
- Drag and drop the event component from the event panel onto the canvas area where you want to display the event. The event component will appear as a visual representation of the event.
- Connect the event component to the Sandbox Reader component."

Congratulations! You've successfully created an event and connected it to the Sandbox Reader component. Your data processing and DS LAB pipeline is taking shape

- Now, let's add the DS Lab component to the canvas for batch processing. Drag and drop the DS Lab component from the ML component palette onto the canvas.
- Select 'batch' as the invocation type and move to the Meta Information tab
- From the execution type dropdown, choose 'Model runner
- Now, select 'churn prediction Project' from the project name dropdown. This project should have been previously created in the DS Lab plugin
- Next, choose 'churn preprocess' dp model from the model name dropdown. This model should have been registered in the DS Lab plugin
- Once you've configured everything, don't forget to save the DS Lab component."

Now, let's create an event and connect it to the component."

- Click on the "Event Panel" icon located in the toolbar. This action will open the event panel, which allows you to manage events.
- Inside the event panel, locate and click on the "Add Event" button. This will initiate the process of adding a new event.
- In the event creation Page, configure all the mandatory fields. Make sure to fill in all the required information accurately.
- Set the partition for the event. The partition is a setting that specifies how the event should be categorized or grouped. In this case, set the partition value to 1.
- Once all the information is filled and the partition is set, click on the "Save" button to create the event. The event will now be added to the system.
- To display the event on the canvas, locate the event component in the event panel. You can usually find it by its icon or name.
- Drag and drop the event component from the event panel onto the canvas area where you want to display the event. The event component will appear as a visual representation of the event.
- let's connect the DS Lab component with the input and output events.

Add Python Script Component

- Drag and drop the Python script component from the scripting section into your canvas or workflow.
- Once the Python script component is added, select the Invocation type as "batch". This means that the script will be executed on the entire DataFrame as a batch operation.
- Go to the meta information section of the Python script component.
- Specify the component name as "DropColumn" to give it a meaningful name.
- Write the script for dropping the 'index' column in the Python script editor:
- In this script, the function func takes a DataFrame df as input and returns a DataFrame df_out with the 'index' column dropped.
- After writing the script, select the function func from the "Start Function" dropdown. This tells the Python script component to use the defined function as the entry point for the script.
- For the input data to the Python script, select "Data frame" from the "In Event Data Type" dropdown, and specify df as the parameter name. This links the input DataFrame to the df parameter of the func function.
- Once all the settings are configured, save the component to apply the changes.

Create and Add Event

- Click on the "Event Panel" icon located in the toolbar. This action will open the event panel, which allows you to manage events.
- Inside the event panel, locate and click on the "Add Event" button. This will initiate the process of adding a new event.
- In the event creation Page, configure all the mandatory fields. Make sure to fill in all the required information accurately.
- Set the partition for the event. The partition is a setting that specifies how the event should be categorized or grouped. In this case, set the partition value to 1.
- Once all the information is filled and the partition is set, click on the "Save" button to create the event. The event will now be added to the system.
- To display the event on the canvas, locate the event component in the event panel. You can usually find it by its icon or name.
- Drag and drop the event component from the event panel onto the canvas area where you want to display the event. The event component will appear as a visual representation of the event.
- let's connect the DS Lab component with the input and output events.

Now, let's add the DS Lab component to the canvas for executing ml model.

- Drag and drop the DS Lab component from the ML component palette onto the canvas.
- Select 'batch' as the invocation type and move to the Meta Information tab

- From the execution type dropdown, choose 'Model runner
- Now, select 'churn prediction Project' from the project name dropdown. This project should have been previously created in the DS Lab plugin
- Next, choose 'churn_model ml model from the model name dropdown. This model should have been registered in the DS Lab plugin
- Click on the "Event Panel" icon located in the toolbar. This action will open the event panel, which allows you to manage events.
- Inside the event panel, locate and click on the "Add Event" button. This will initiate the process of adding a new event.
- In the event creation Page, configure all the mandatory fields. Make sure to fill in all the required information accurately.
- Set the partition for the event. The partition is a setting that specifies how the event should be categorized or grouped. In this case, set the partition value to 1.
- Once all the information is filled and the partition is set, click on the "Save" button to create the event. The event will now be added to the system.
- To display the event on the canvas, locate the event component in the event panel. You can usually find it by its icon or name.
- Drag and drop the event component from the event panel onto the canvas area where you want to display the event. The event component will appear as a visual representation of the event.
- let's connect the DS Lab component with the input and output events.

Now, let's add the DS Lab component to the canvas for run or execute enrichment script.

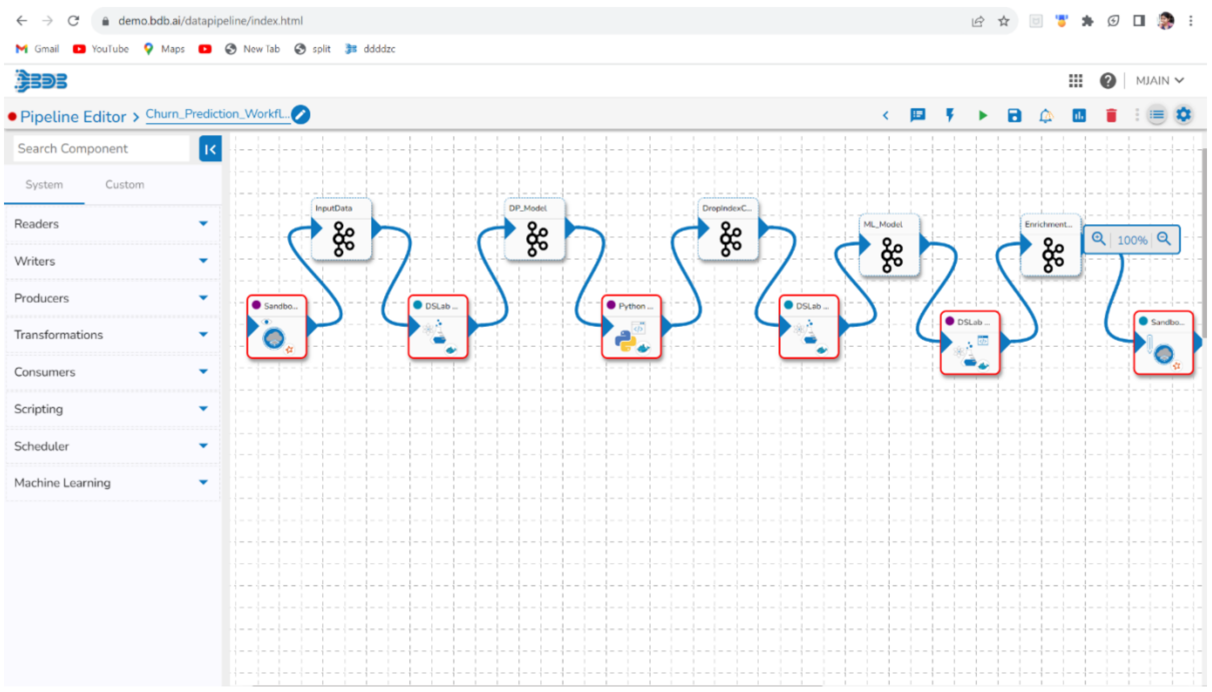
- Drag and drop the DS Lab component from the ML component palette onto the canvas.
- Select 'batch' as the invocation type and move to the Meta Information tab
- From the execution type dropdown, choose Script runner
- Now, select 'churn prediction Project' from the project name dropdown. This project should have been previously created in the DS Lab plugin
- Next, choose churn_pred_util script from the Script name dropdown. This model should have been registered in the DS Lab plugin
- Function type as data frame
- Select func from the start function dropdown
- Let's pass Secrets credentials in input data section
- orig_table_name churn_data
- host @ENV.DS_CH_HOST
- port @ENV.DS_CH_TCP_PORT
- database @ENV.DS_CH_DB_DEVELOPMENT
- user @ENV.DS_CH_USER_DEVELOPMENT
- save the component

- Click on the "Event Panel" icon located in the toolbar. This action will open the event panel, which allows you to manage events.
- Inside the event panel, locate and click on the "Add Event" button. This will initiate the process of adding a new event.
- In the event creation Page, configure all the mandatory fields. Make sure to fill in all the required information accurately.
- Set the partition for the event. The partition is a setting that specifies how the event should be categorized or grouped. In this case, set the partition value to 1.
- Once all the information is filled and the partition is set, click on the "Save" button to create the event. The event will now be added to the system.
- To display the event on the canvas, locate the event component in the event panel. You can usually find it by its icon or name.
- Drag and drop the event component from the event panel onto the canvas area where you want to display the event. The event component will appear as a visual representation of the event.
- let's connect the DS Lab component with the input and output events.

Now it's time to select the appropriate Writer component. Let's add the Sandbox Writer component from the Writer section and configure it

- Drag and drop the Sandbox Writer component onto the canvas
- "Select 'Realtime' as the invocation type and move to the Meta Information tab."
- "In the Meta Information tab, choose 'network' as the storage type."
- "Next, specify the sandbox file name where you want to write the data."
- "Select 'csv' as the file type for writing the data."
- "Now, choose 'Overwrite mode' from the save mode dropdown to ensure the data is overwritten each time the pipeline runs."
- "Once you've configured the component, click on the save button to save your settings. "Now, connect the output event of the DS Lab component to the input of the Sandbox Writer component."

Congratulations! You've successfully added and configured the Sandbox Writer component. Your end-to-end data processing and DS Lab pipeline is now complete, ready to process data in realtime and write the results to the specified file. You can now run the pipeline to start the data processing and ds lab tasks."



After configuring and setting up the Pipeline Workflow, it's time to Update and activate the pipeline.

- Locate "Update Pipeline" icon in the toolbar and Clicks on it
- Now, Click on the 'Activate Pipeline button. This will Start the execution of the Pipeline and start the data processing.
- After activating the Pipeline, navigate to the logs and advance Log section, Look for the Log Panel and click on it to access the advanced logs for detailed information.
- Within the Log Panel, you'll see the pods associated with each component. Pods are containers that hold the execution environment for the Pipeline.
- Check if the pods for each component have come up and are running. This indicates that the components are successfully deployed and ready to execute their tasks.
- To view the specific logs for each component, click on the corresponding pod or log entry. The logs will provide detailed information about the execution and any potential errors or issues encountered during the process.

In this workflow, we are building an end-to-end data processing and DS Lab pipeline to read churn data from the sandbox, apply a model created in the DS Lab plugin, and then write the processed data back to the sandbox.

In our end-to-end data processing and DS Lab pipeline for churn data analysis, we embark on a journey to extract valuable insights from the churn data residing in the sandbox. The first step involves seamlessly connecting to the sandbox and retrieving the churn data, which will serve as the foundation for our entire analysis. Leveraging the powerful features of DS Lab, we ensure a smooth and efficient data retrieval process. With the churn data at our disposal, we move on to the next step, where we explore the predictive capabilities of the DS Lab model. This carefully crafted model is designed specifically for churn prediction and plays a vital role in helping us make well-informed decisions. We eagerly run the churn data through our model, witnessing its accurate predictions, identifying potential churners, and gaining valuable insights into customer behavior. However, our journey doesn't stop there. In the subsequent step, we delve into the realm of data processing and enrichment. Employing essential data processing techniques, we meticulously clean and prepare the churn data, ensuring its quality and reliability for further analysis. Additionally, we enrich the data with relevant information, harnessing the power of additional variables to enhance the predictive power of our model. With the processed and enriched churn data now at its best, the final step involves writing it back to the sandbox. This secure storage ensures that the valuable data can be accessed for further exploration, analysis, or even reporting purposes. Our end-to-end pipeline enables us to seamlessly progress from data retrieval to prediction and data enhancement, culminating in a powerful and comprehensive churn analysis to drive strategic decision-making and customer retention efforts.

Once your data is successfully written, let's move on to the next steps.

- locate and select data center plugin from the app menu
- Inside the Data Center Plugin, navigate to the "Sandbox" section. This is where we'll create a sandbox to store our data.

- let's proceed to create a datastore. A datastore is like a repository where you can access and manage your data.
- In the Business Story section, select the appropriate visualization type (bar graphs, line charts, pie charts, etc.) based on your data and what you want to represent.

After completing your workflow and ensuring that all your tasks are accomplished, it's essential to deactivate the pipeline and Data Science (DS) lab to save resources and prevent any unnecessary usage.

Thankyou

BDB 8.2: Data Science Lab Workflow – 4

Create a Sentiment Model within the DS Lab Notebook and Register the Model as an Api.

In Workflow 4, we will explore the effectiveness and ease of creating a Sentiment Analysis model within the DS Lab Notebook. Additionally, we'll delve into the API Client registration features of the Admin Module to register the model as an API. Afterward, we will test the model's response for input data by sending an API request within the notebook.

In this workflow, we'll cover the following key steps:

- DS Project Creation
- Create Sentiment Analysis Model
- Register the model as an API and utilize the admin module for API Client Registration.
- Test the model as an API within the DS Lab notebook by sending an API request to obtain the sentiment response.

Note - "When parking your vehicle, be sure to turn off both the engine and headlights."

To optimize resource consumption, it's essential to deactivate & unregister any active projects, pipelines, or models when you're finished with them. This step releases resources for other tasks and helps prevent potential issues.

Let's take care of this step first, then move on to the rest of your tasks.

For Dataset and Sample Python Notebook: Click Here to Download

DS Lab Project Creation

Here, we will cover the step-by-step process of creating a new Data Science Lab Project and performing various Data Science tasks on the data.

To Begin,

- Firstly, create a DS lab project.
- Provide the project name and description.
- Choose the algorithm as classification.
- Specify the environment and resources.
- Set the idle shutdown time.
- In the external library section, include the names of the libraries to be used, such as spaCy and tqdm.

- This will eliminate the need to download these libraries within the notebook.
- Finally, save your project.

Create Sentiment Analysis Model

Here, we will cover the step-by-step process of creating a Sentiment Model on top of 'Musical_instruments_reviews' Dataset.

To Begin,

- Activate your project and open it.
- Click on "Upload Notebook" to choose the notebook from your local machine or Else create Notebook and write complete code.
- Provide a name to Notebook.
- Load the data and create a dataframe by clicking on "Dataset" and selecting the dataset or uploading your data.
- Use the checkbox to automatically populate the code for reading and loading data into the dataframe.
- Map the sentiment numbers to sentiment values and drop unwanted columns from the dataframe.
- Download the English language model 'en_core_web_sm' provided by spaCy and wait for the download to complete.
- Create a subset for model training by using the df.iloc method and copying the first 200 rows of data.
- The 'text_col' variable holds the name of the column, i.e., 'reviewText', which contains the text data.
- Import the TfidfVectorizer class from the sklearn.feature_extraction.text module.
- Create an instance of TfidfVectorizer and store it in the 'tfidf' variable.
- Apply the fit_transform method to the text input data column and store it in the variable 'X_tfidf'.
- Use the provided boilerplate code or write custom code for a logistic regression classification model.
- Import the required libraries from sklearn.
- The sentiment column from the data is the target variable.
- Split the data into training and testing sets using train_test_split.
- Train the logistic regression model using the fit method.
- Evaluate the model's performance on the test set using the score method.
- Make predictions on the data and print classification reports for both the training and test sets.

- Save the model using the built-in BDB DSLab library.
- Click on the three dots on the right-hand side of the cell and choose "Save Model" to auto-populate the code.
- Specify the model, model name, and model type for saving the model.
- Execute the code and save the model.

Model as API and API Client Registration

Here, we will cover the step-by-step process of Registering the model as an API and utilize the admin module for API Client Registration.

To Begin,

- Register the model as an API by navigating to the model tab and selecting the desired model for registration.
- Once selected, proceed to register it as an API by clicking the corresponding option.
- Provide necessary information, such as instance and resources, and save the registration details.
- For API client registration, this task is restricted to administrators only and not available to regular users.
- Users can use the information provided in the attached video or document for experimentation purposes.
- Access the admin plugin to initiate API client registration.
- Two options will be presented: internal and external. Choose "internal" for the current task.
- Enter the client's name and email address, where the client will receive API information.
- Provide the App name, request per hour, and request per day limits for the client's API usage.
- Select the specific model (previously registered) for the API client.
- Save the registration details to complete the process.
- To share API credentials with the client, you can send an email by clicking on the icon representing the secret ID and secret key.
- Alternatively, you can click on the edit icon to access and provide the necessary details directly.

Note: To manage resources efficiently, you may consider unregistering any unwanted models, as they consume resources.

Test Model As API

Here, we will cover the step-by-step process to test the model as an API within the DS Lab notebook by sending an API request to obtain the sentiment response.

To Begin,

- Open DS lab and navigate to your notebook for testing the model as an API.
- Create a variable, 'api_test_input', to store the input data for the model.
- Import the 'requests' and 'json' modules to work with API requests and JSON data.
- Create a URL by appending the model name with the '.dill' extension to the base URL.
- Set up the payload with the required input data for the API request.
- Ensure you have the necessary header details like client ID, client secret, and app name, which can be found in the API client registration section or copied from the email received during registration.
- Execute the code to send the API request and receive the output response.
- Print the sentiment response obtained from the API.
- Remember to save the notebook after completing the testing process to retain the changes and results.

Hope you will be able to create your Data Science Workflows. **Thank you!**

BDB 8.2: Data Science Lab Workflow – 5

Workflow 5 empowers us to automate the machine learning process, integrate data preparation techniques, and reveal valuable insights from our Super Market data. Step into the world of AutoML and unlock the full potential of your data. Introducing the power of Automated Machine Learning in simplifying and accelerating the application of machine learning algorithms to real-world problems. With Workflow 5, you have the ability to effortlessly create AutoML experiments specifically tailored for Super Market data, harnessing the capabilities of both classification and regression algorithms. Our Super Market dataset holds a wealth of valuable information, encompassing product sales, customer demographics, and market trends. By leveraging Workflow 5, you can tap into this data goldmine and uncover valuable insights that drive informed decision-making.

In this workflow, you will upload your Super Market dataset into the AutoML platform's sandbox environment. After applying data preparation techniques to clean the dataset, you'll create an AutoML experiment using both classification and regression algorithms. Once the experiment is completed, a comprehensive report will be generated. You can then analyze the performance of the models, explore model explainer dashboards to gain insights into the models' predictions and behavior, and utilize the dataset explainer to understand patterns and relationships within the Super Market dataset. This workflow enables you to unlock valuable insights and make informed decisions based on the analysis of the models and data. you can upload your dataset using both the Data Center plugin and DS Lab plugin to provide flexibility and convenience.

Note - "When parking your vehicle, be sure to turn off both the engine and headlights."

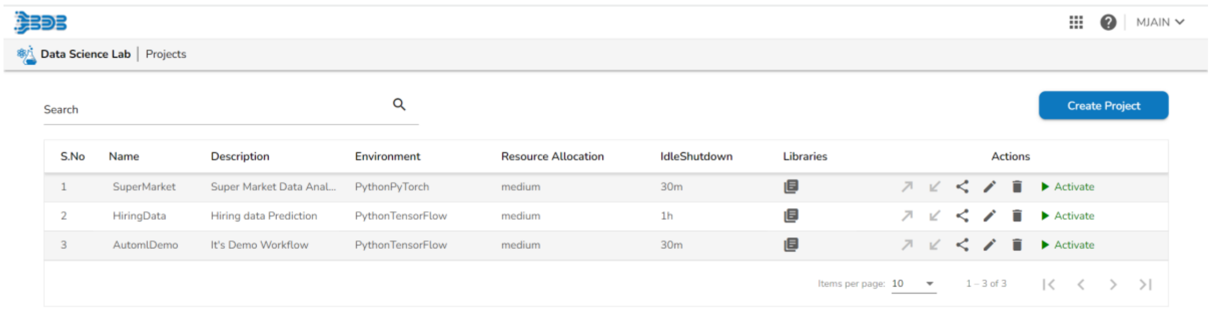
To optimize resource consumption, it's essential to deactivate & unregister any active projects, pipelines, or models when you're finished with them. This step releases resources for other tasks and helps prevent potential issues.

For Dataset: [Click Here](#) to Download

Let's take care of this step first, then move on to the rest of your tasks

To create an AutoML workflow, follow these steps

- After accessing the home screen of the platform, select the "DS Lab" model from the app menu. This will take you to the home page of the Data Science Lab.
- On the home page, you will see a list of projects. Each project will have information such as its name, description, environment, resource allocation type, libraries, and various action items like pushing and pulling the project from version control systems, sharing the project, editing, deleting, and activating it.



S.No	Name	Description	Environment	Resource Allocation	IdleShutdown	Libraries	Actions
1	SuperMarket	Super Market Data Anal...	PythonPyTorch	medium	30m		▶ Activate
2	HiringData	Hiring data Prediction	PythonTensorFlow	medium	1h		▶ Activate
3	AutomlDemo	It's Demo Workflow	PythonTensorFlow	medium	30m		▶ Activate

Items per page: 10 1 – 3 of 3

- Select the desired project from the list. This will take you to the project's main page, where you will find multiple tabs such as Notebook, Dataset, Utility, Model, and AutoML.
- To work on the AutoML workflow, navigate to the "Dataset" tab. This tab is where you can manage and explore the datasets associated with your project.
- In the Dataset tab, you will see a list of uploaded datasets. Each dataset will have information such as its type and various actions you can perform, such as previewing the data, generating a data profile, creating an experiment, deleting the dataset, and data preparation.

- Select the appropriate dataset for your Super Market project. If the dataset is not available, you can follow these steps.
- On the right-hand side, there is an "Add Dataset" button. Click on it to navigate to the "Add Dataset" page.
- In the "Add Dataset" page, select the "Data Sandbox" option as the data source. This will list all the existing files in the sandbox. If you want to add a new dataset to the sandbox, click the "Upload" button, which will take you to the "Upload Data Sandbox" page.

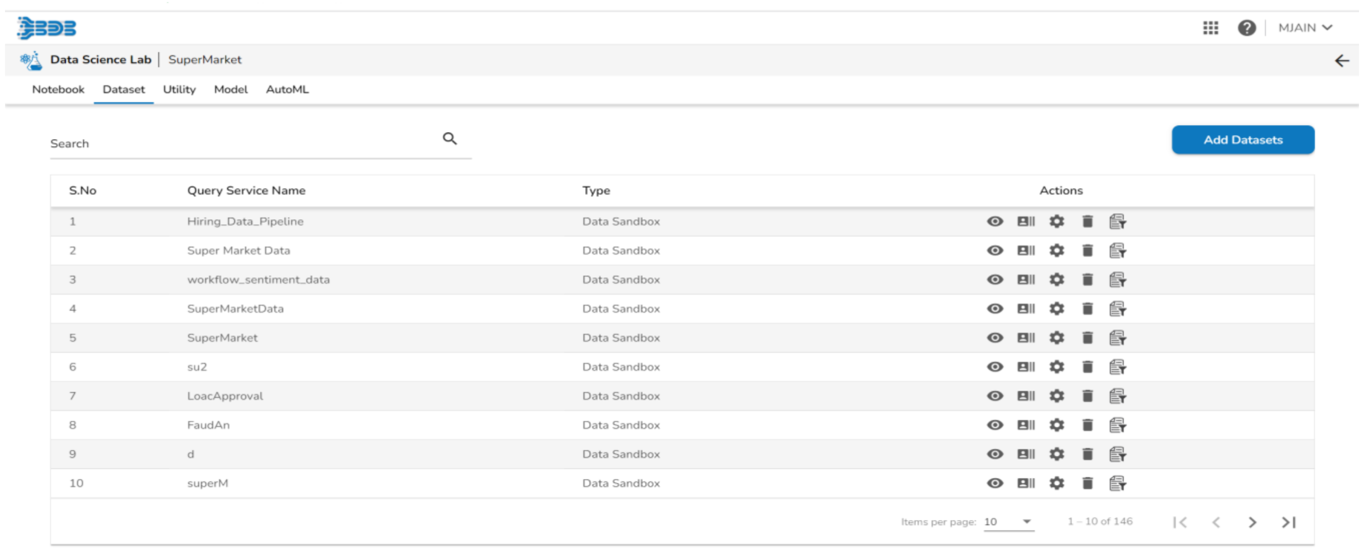
Data Sandbox

In the "Upload Data Sandbox" page, let's give the sandbox a name. Choose a name that will help you identify it later. For example, let's name it SuperMarketData and Provide an appropriate description if needed. This can help provide more context or details about the purpose of the sandbox let's specify Classification Super market Experiment

Next, you'll need to choose your data file. Look for an option to upload or select a data file for the sandbox. Before uploading the file, ensure that it follows the specified format:

- The first row in the file should contain the column headers.
- The headers should not have spaces and should be a single word or two words concatenated by an underscore (_).
- The headers should not contain any special characters like %, #, $, @, *, etc.
- The header should have at least one alphabet and should not consist solely of numerals.
- All cells in a column should have a single data type.
- The header should not use single or double quotes, dots, brackets, or hyphens.
- Files with utf8 encoding are not supported for data store creation.
- Once you have selected the file and ensured it follows the required format, click the "Save" button to upload the dataset to the sandbox.
- File is uploaded

By following these steps, you can add a dataset to the sandbox in the Data Science Lab, which will allow you to proceed with the AutoML workflow using that dataset.



S.No	Query Service Name	Type	Actions
1	Hiring_Data_Pipeline	Data Sandbox	
2	Super Market Data	Data Sandbox	
3	workflow_sentiment_data	Data Sandbox	
4	SuperMarketData	Data Sandbox	
5	SuperMarket	Data Sandbox	
6	su2	Data Sandbox	
7	LoacApproval	Data Sandbox	
8	FaudAn	Data Sandbox	
9	d	Data Sandbox	
10	superM	Data Sandbox	

Items per page: 10 1 – 10 of 146

After adding the dataset to the sandbox, follow these steps:

Click on the checkbox next to the file you have just uploaded. This will select the dataset for further actions.

On the right side of the page, you will see various action items for the selected dataset.

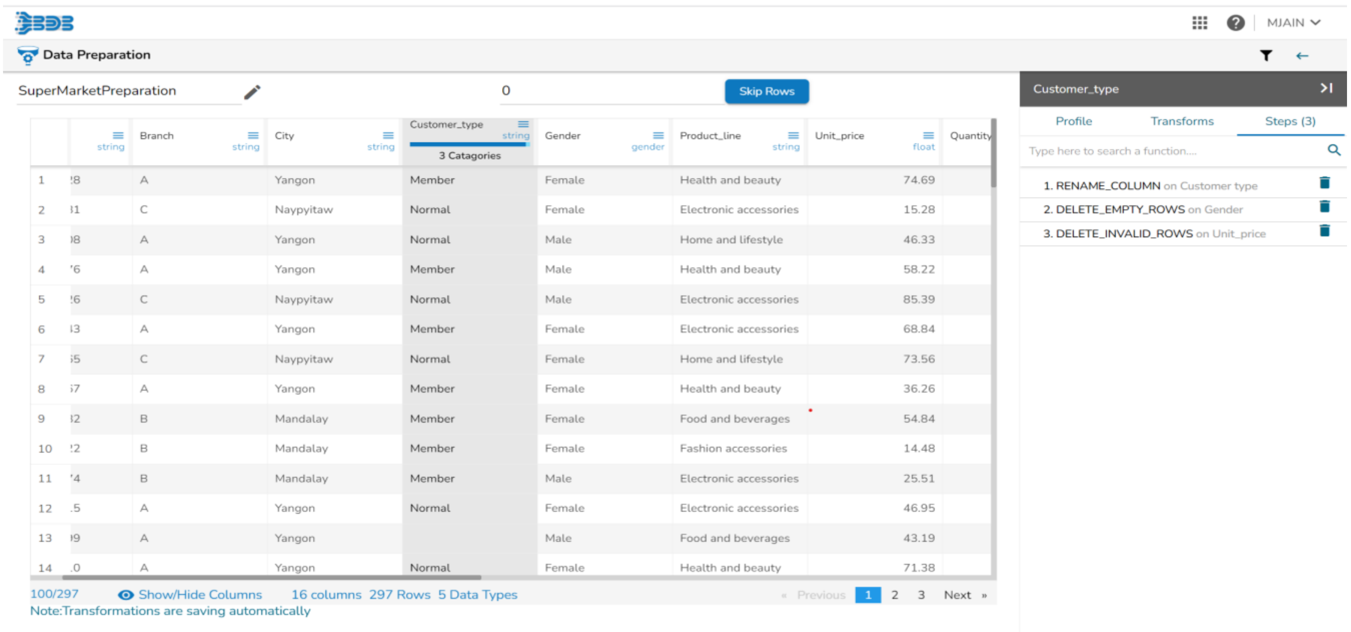
- Preview: Clicking on this option will display a sample of the actual data, allowing you to understand the data values and structure better.
- Data Profile: Selecting this option will open the Data Profile page, where you can visualize detailed information about the dataset. This includes data set information, variable types, warnings, variables, correlation chart, missing values, and a data sample.
- Create Experiment: This option allows you to create an AutoML experiment using the selected dataset. You can proceed with configuring the experiment to apply classification and regression algorithms, as mentioned earlier in the workflow.
- Data Preparation: Data preparation is an essential step in data analysis and machine learning. By selecting this option, you can perform data cleaning, transformation, and feature engineering to ensure the data's accuracy, completeness, consistency, and relevance for analysis.
- Delete Dataset: If you wish to remove the dataset from the sandbox, you can use this option to delete it.

Create Preparation

Data Preparation is used to clean the data. Let's get started with the data preparation process

- Select Data Preparation Icon. will navigate to the Data preparation home page

- Here on the Data Preparation home page, you can see your complete dataset displayed in a grid form. The Data Preparation Plugin automatically profiles the data,
- Providing valuable insights into its characteristics and detecting any anomaly data. You can also view the data profiling details
- On the right-hand side, you'll find the selected column's profile. Here, you can explore various details such as charts, information, and patterns associated with the selected column
- All the Transformations will appear inside transform tab
- Now, let's start preparing and cleaning the data.
  - To remove the empty cells in the Gender' column, just select Gender column and navigate to the 'Transforms' tab and search for the 'Delete Rows with Empty Cells' transform. Click on it to remove all the empty rows from the Gender column". You can see that the empty rows in the Gender' column have been removed
  - Next, let's perform the Delete rows with invalid cell transformation on the Unit Price' column.
- Select the column and search for the Delete rows with invalid cell ' transform. Click on it to Delete invalid cell from the Unit Price ' column.
- Great! The Invalid cell in the Unit Price ' column have been successfully removed."
- Now, let's Rename the Customer type column from the dataset. Simply select the column and search for the Rename Column' transform.
- Click on it to and rename it let me specify customer_type and submit it. Perfect! The customer type column has been Renamed.
- You can see that all the performed transforms are recorded in the 'Steps' section. This helps you keep track of the changes made to the dataset.
- Now, let's rename the preparation for identification purposes. Simply click on the edit icon and give it a new name, such as 'SuperMarketPreparation



1. RENAME_COLUMN on Customer type
2. DELETE_EMPTY_ROWS on Gender
3. DELETE_INVALID_ROWS on Unit_price

	string	Branch string	City string	Customer_type string 3 Catagories	Gender gender	Product_line string	Unit_price float	Quantity
1	:8	A	Yangon	Member	Female	Health and beauty	74.69	
2	:1	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	
3	)8	A	Yangon	Normal	Male	Home and lifestyle	46.33	
4	'6	A	Yangon	Member	Male	Health and beauty	58.22	
5	:6	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	
6	:3	A	Yangon	Member	Female	Electronic accessories	68.84	
7	;5	C	Naypyitaw	Normal	Female	Home and lifestyle	73.56	
8	;7	A	Yangon	Member	Female	Health and beauty	36.26	
9	:2	B	Mandalay	Member	Female	Food and beverages	54.84	
10	:2	B	Mandalay	Member	Female	Fashion accessories	14.48	
11	'4	B	Mandalay	Member	Male	Electronic accessories	25.51	
12	.5	A	Yangon	Normal	Female	Electronic accessories	46.95	
13	)9	A	Yangon		Male	Food and beverages	43.19	
14	.0	A	Yangon	Normal	Female	Health and beauty	71.38	

100/297 Show/Hide Columns 16 columns 297 Rows 5 Data Types « Previous 1 2 3 Next »

Note:Transformations are saving automatically

Great! The preparation has been renamed to 'SuperMarketPreparation Click on the back icon, and the preparation will be automatically saved and exported to different plugins, such as the Data Pipeline or AutoML/DS Lab

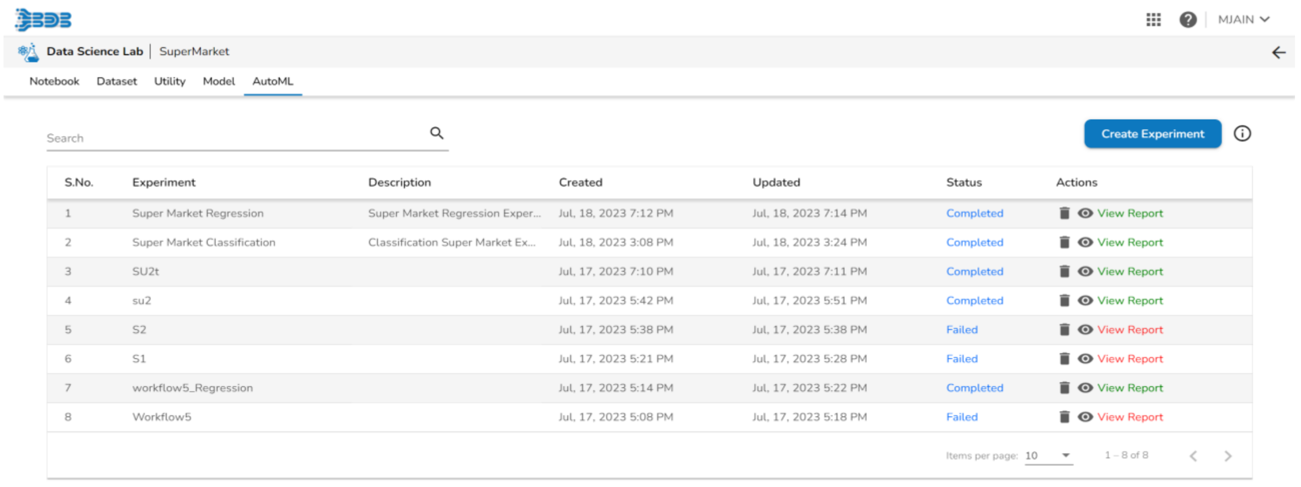
AutoML Experiment Creation

Data scientists can create and manage their experiments effortlessly. Let's take a look at how to create a new AutoML experiment.

- To begin, click on the "Create Experiment" icon and select the dataset you want to work with. This will serve as the foundation for your experiment
- Now, let's configure the experiment-specific details. Start by providing a name for your experiment. This will help you easily identify it later. let me give Supermarket Classification. You can also add an optional description if you wish. let me specify classification on the "Production Line" Field
- Great! Next, select a target column for your experiment. In this case, we'll choose "Production Line" as the target column.
- Excellent choice! Now, let's move on to the Data Preparation step. From the dropdown menu, select "Preparation," and then choose the "SuperMarketPreparation" option.
- With the configuration complete, you'll now be redirected to the "Experiment Type" tab.
- Here, you can choose the prediction model based on your needs. Since we selected "Production Line" as the target column, we'll choose the Classification prediction model
- Perfect! Now that you've completed the necessary steps, click "Done" to finalize your experiment setup. Congratulations! You have successfully created a new AutoML experiment. You will receive a notification confirming its creation.

When you navigate to the AutoML tab, you will see a list of all your experiments. Your newly created experiment will appear at the top.

- Each experiment has a status associated with it, indicating its progress and outcome. Initially, a new experiment has the status of "Started."
- As the model training progresses, the status changes to "Running." You will receive a notification to keep you informed.
- Once the model training is successfully completed, the status changes to "Completed," indicated by a green color.



S.No.	Experiment	Description	Created	Updated	Status	Actions
1	Super Market Regression	Super Market Regression Exper...	Jul, 18, 2023 7:12 PM	Jul, 18, 2023 7:14 PM	Completed	View Report
2	Super Market Classification	Classification Super Market Ex...	Jul, 18, 2023 3:08 PM	Jul, 18, 2023 3:24 PM	Completed	View Report
3	SU2t		Jul, 17, 2023 7:10 PM	Jul, 17, 2023 7:11 PM	Completed	View Report
4	su2		Jul, 17, 2023 5:42 PM	Jul, 17, 2023 5:51 PM	Completed	View Report
5	S2		Jul, 17, 2023 5:38 PM	Jul, 17, 2023 5:38 PM	Failed	View Report
6	S1		Jul, 17, 2023 5:21 PM	Jul, 17, 2023 5:28 PM	Failed	View Report
7	workflow5_Regression		Jul, 17, 2023 5:14 PM	Jul, 17, 2023 5:22 PM	Completed	View Report
8	Workflow5		Jul, 17, 2023 5:08 PM	Jul, 17, 2023 5:18 PM	Failed	View Report

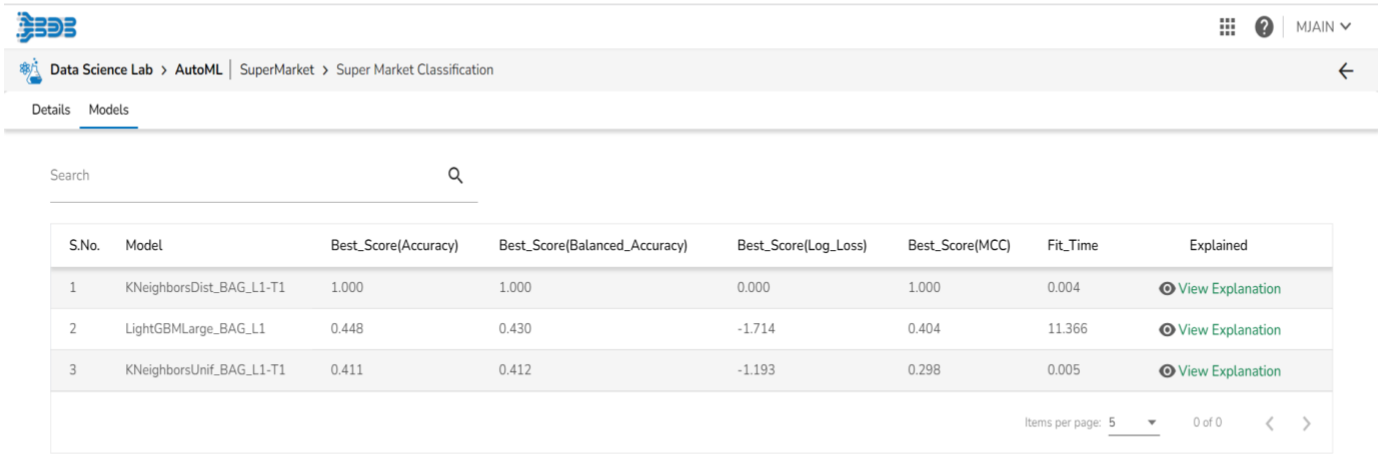
Items per page: 10 1 – 8 of 8

- However, if any issues arise during training, the status will change to "Failed," and it will be indicated in red.
- Remember, you have the option to delete an experiment or view its detailed report.
- Use the "Delete" button to remove an experiment from the list, and click "View Report" to access a comprehensive analysis of the experiment's results.
- Once your experiment is completed, navigate to the experiment dashboard or interface. Look for the completed experiment that you want to analyze. Click on it to open the experiment details.
- Within the experiment details, you will find several tabs. Click on the 'Details' tab to access a comprehensive overview of the experiment and its trained model.

View Report

- In the 'Details' tab, you will see the 'Recommended Model' section. This section highlights the best suitable model based on the evaluation metric. Take note of the 'Model Name' and 'Model Score' which represent the name of the model and its score respectively.

- Further down in the 'Details' tab, you will find the 'Metric Value' section. This section displays the specific metric that was used to evaluate and select the model during the experiment. It gives you an understanding of how the recommended model performed.



Data Science Lab > AutoML | SuperMarket > Super Market Classification

Details | Models

S.No.	Model	Best_Score(Accuracy)	Best_Score(Balanced_Accuracy)	Best_Score(Log_Loss)	Best_Score(MCC)	Fit_Time	Explained
1	KNeighborsDist_BAG_L1-T1	1.000	1.000	0.000	1.000	0.004	View Explanation
2	LightGBMLarge_BAG_L1	0.448	0.430	-1.714	0.404	11.366	View Explanation
3	KNeighborsUnif_BAG_L1-T1	0.411	0.412	-1.193	0.298	0.005	View Explanation

Items per page: 5 0 of 0

- You will also find other information such as the 'Created On' date and the 'Run Summary'. The run summary provides basic experiment and model details including the task type, primary metric, model status, and the name of the creator.

- To gain further insights, navigate to the 'Models' tab. Here, you will see a list of the top three models based on their metric scores. These models are generated from a total of 30 runs, including models from AutoGluon, Grid Search, and Random Search.

- Within the 'Models' tab, you have the option to 'View Explanation' for each model. Clicking on this option redirects you to a page where you can access various details about the selected model.

- The 'Model Summary' provides essential information about the selected model. Here, you will find details such as the algorithm name, model status, creation date, start date, duration, and performance metrics. This summary gives you an overview of the selected model's characteristics.

- Within the model explanations, feature importance analysis is a crucial aspect. This analysis helps determine which features had the biggest impact on the model's predictions. By examining feature importance, you gain insights into the key factors influencing the model's decision-making process.

'Classification Stats' tab provides various statistics regarding the classification model. Here, you can access a range of performance metrics that evaluate the model's accuracy and effectiveness. These metrics offer insights into how well the model performs in classifying positive and negative instances.

Global Cutoff

Within the 'Classification Stats' tab, you can set a global cutoff. This cutoff is a threshold that determines the classification of instances into positive or negative classes. By setting the cutoff, you can control the balance between false positives and false negatives, optimizing the model's performance.

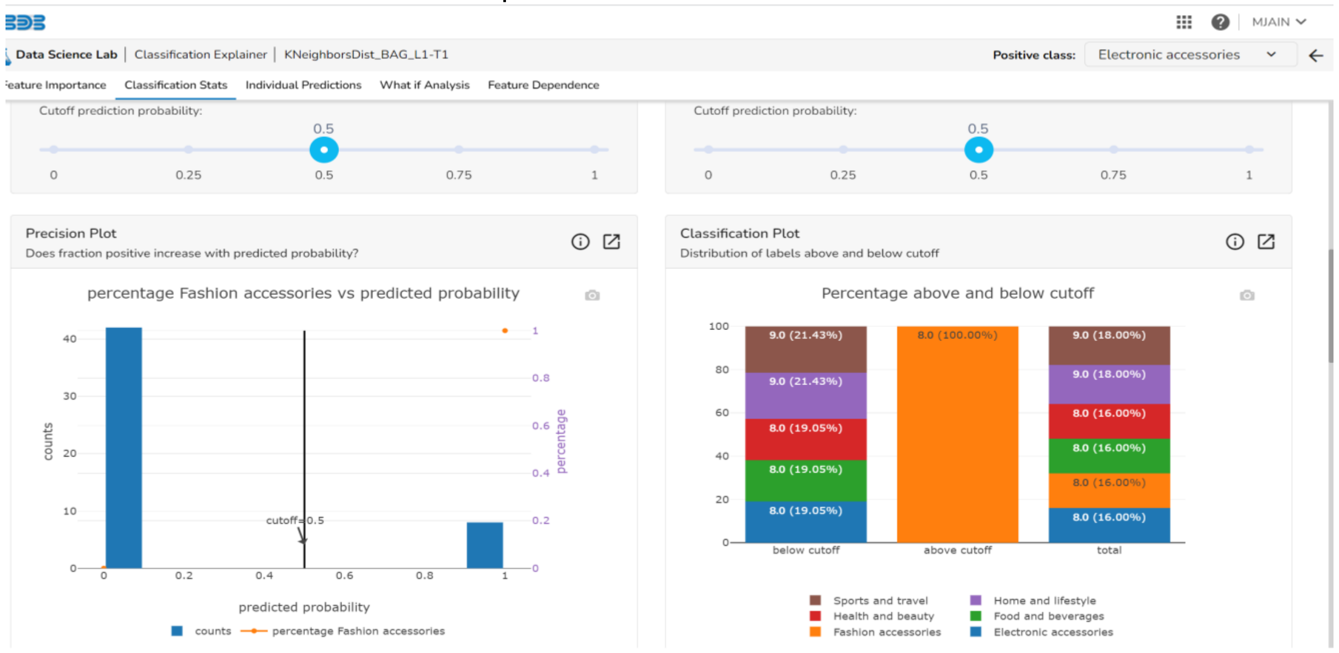
Model Performance Metrics

The 'Model Performance Metrics' section displays a list of various performance metrics. These metrics provide valuable insights into the model's performance, including accuracy, precision, recall, F1 score, and more. Reviewing these metrics helps you assess the model's strengths and weaknesses in classification tasks.

Confusion Matrix

The confusion matrix presents a visual representation of the model's performance in classifying instances. It shows the number of true negatives, true positives, false negatives, and false positives. By examining the confusion matrix, you can understand the costs associated with

misclassifications and select an optimal cutoff.



Precision Plot

The precision plot illustrates the relationship between the predicted probability of a record belonging to the positive class and the percentage of observed records in the positive class. This plot helps assess the model's calibration and its ability to accurately predict the positive class.

Classification Plot

The classification plot displays the fraction of each class above and below the selected cutoff. It provides insights into how the model's predictions are distributed between the positive and negative classes based on the chosen threshold.

ROC AUC Plot

The ROC AUC plot is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds. It helps assess the model's performance in distinguishing between positive and negative instances across different thresholds.

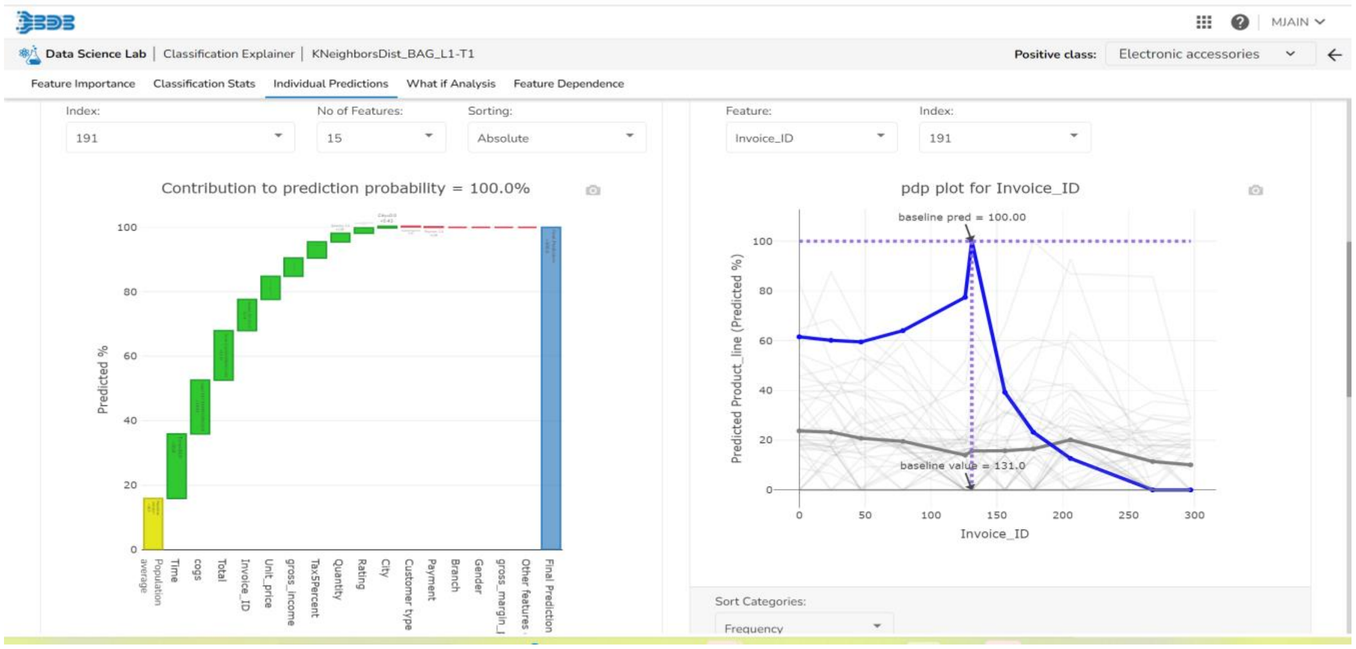
PR AUC Plot

The PR AUC plot shows the trade-off between precision and recall in a single plot. It provides insights into how the model's precision and recall change with varying classification thresholds.

Lift Curve and Cumulative Precision

The lift curve chart displays the percentage of positive classes when selecting observations with scores above the cutoff compared to random selection. It helps evaluate the model's performance compared to random selection.

Individual Prediction: The individual prediction section displays the predicted probability for each target label. It provides insights into how the model assigns probabilities to different classes for a specific observation.



Contributions Plot

The contributions plot shows the contribution that each feature has provided to the prediction for a specific observation. These contributions, starting from the population average, add up to the final prediction. This plot helps explain how each prediction is built up from the individual features in the model.

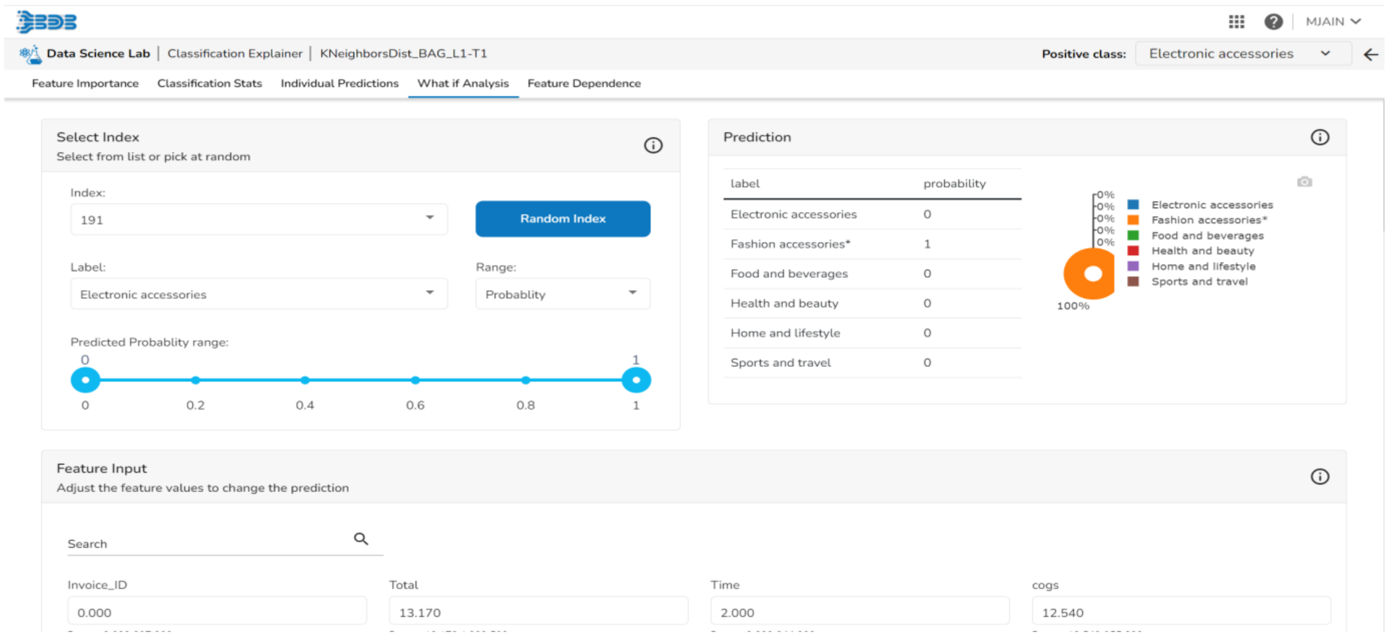
Partial Dependence Plot

The partial dependence plot (PDP) shows how the model prediction would change if you change a particular feature while keeping other features constant. It provides insights into the relationship between a specific feature and the model's predictions. The average effect is shown in grey, and the effect of changing the feature for a single record is shown in blue.

Contributions Table

The contributions table shows the contribution each individual feature has had on the prediction for a specific observation. These contributions, starting from the population average, add up to the final prediction. This table allows you to explain how each individual prediction is built up from the features in the model.

What If Analysis: The What If Analysis allows stakeholders to understand the potential consequences of different scenarios or decisions. In this analysis, you can change the values of selected variables to see how the outcome would change. It helps identify the sensitivity of the outcome to different inputs and which variables are most important.



Select Index
Select from list or pick at random

Index: 191

Label: Electronic accessories

Range: Probablity

Predicted Probablity range: 0 – 1

Prediction

label	probability
Electronic accessories	0
Fashion accessories*	1
Food and beverages	0
Health and beauty	0
Home and lifestyle	0
Sports and travel	0

🟦 Electronic accessories
🟧 Fashion accessories*
🟩 Food and beverages
🟥 Health and beauty
🟪 Home and lifestyle
🟫 Sports and travel

Feature Input
Adjust the feature values to change the prediction

Invoice_ID	Total	Time	cogs
0.000	13.170	2.000	12.540

Feature Input

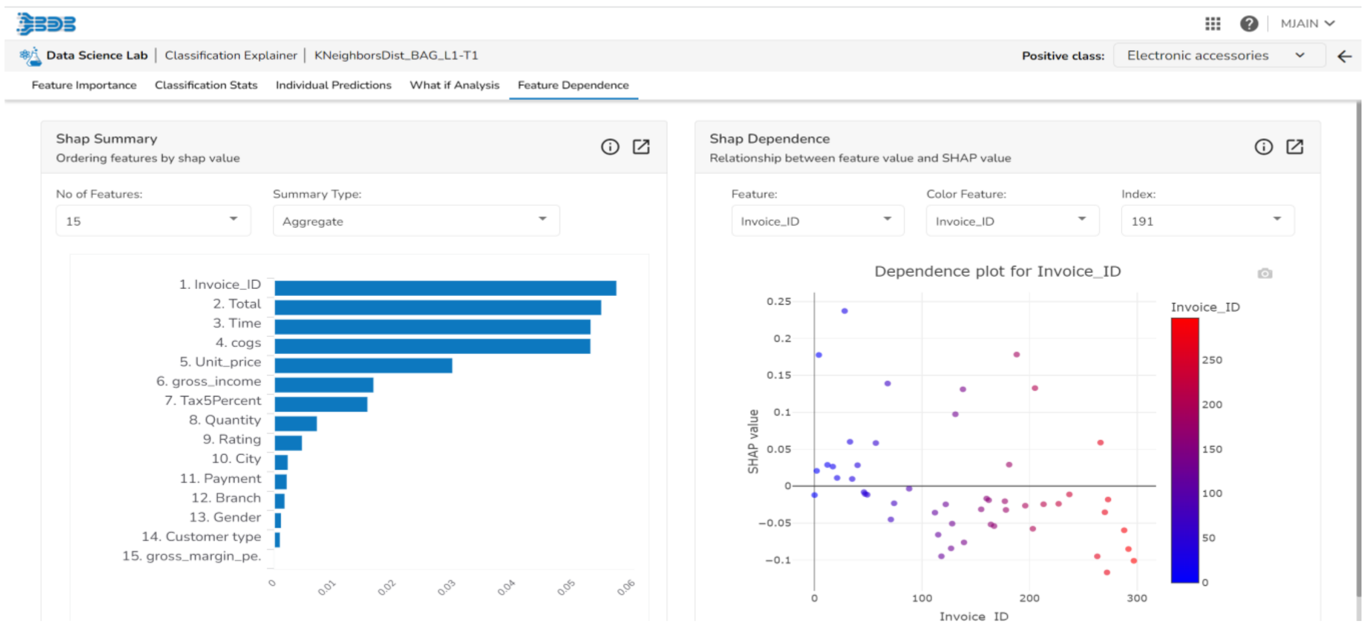
Within the What If Analysis, you can adjust the input values to see predictions for different scenarios. This feature allows you to explore how changing input values affects the model's predictions.

Contribution & Partial Dependence Plots

In the What If Analysis, analysts typically start with a baseline scenario and identify variables that may impact the outcome. Contribution and partial dependence plots help visualize the effects of changing these variables on the model's predictions.

Feature Dependence

The feature dependence analysis explores the relationship between feature values and their impact on predictions. This analysis allows you to investigate how the model uses features in line with intuitions or learn about the relationships the model has learned between input features and predicted outcomes.



Positive class: Electronic accessories

No of Features: 15

Summary Type: Aggregate

Feature: Invoice_ID

Color Feature: Invoice_ID

Index: 191

The Shap summary summarizes the Shap values per feature. It provides an aggregate display, showing the mean absolute Shap value per feature. This summary helps understand the overall impact of each feature on the model's predictions.

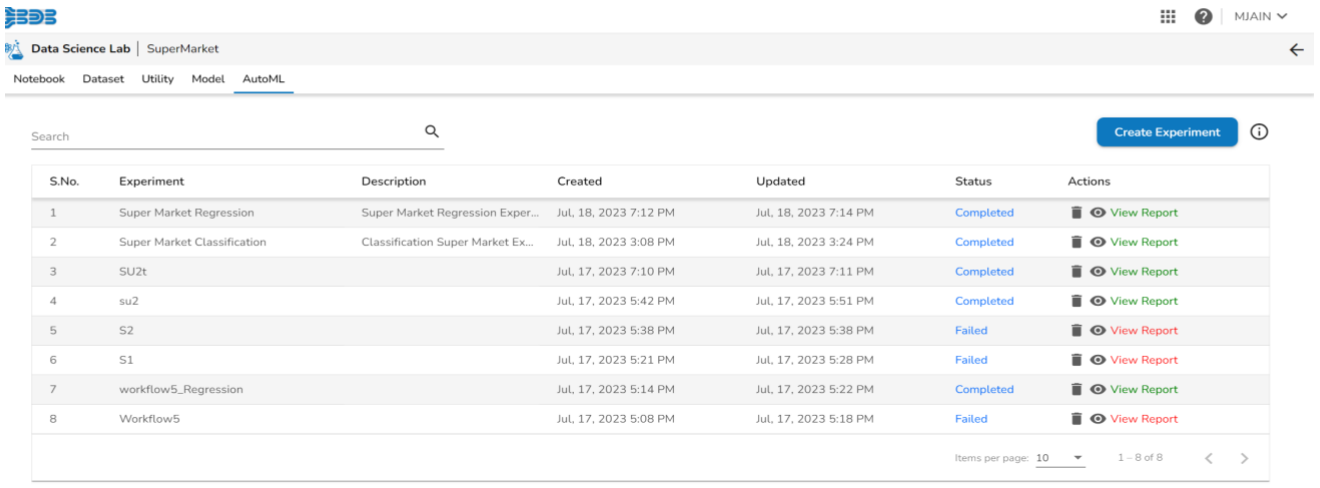
Shap Dependence

The Shap dependence plot displays the relationship between feature values and Shap values. It allows you to investigate the general relationship between feature values and their impact on predictions. This plot helps you understand how the model uses features and their influence on the predicted outcome.

Regression Automl Experiment

Let’s Create an AutoML Regression Experiment for Supermarket Sales Prediction. Data scientists can easily create and manage their experiments with AutoML. Let's see how to create an AutoML Regression experiment.

- To begin, click on the 'Create Experiment' icon and select the dataset you want to work with. This dataset will serve as the foundation for your experiment.
- Now, let's configure the experiment-specific details. Start by providing a name for your experiment, such as 'Supermarket Regression.' You can also add an optional description to easily identify it later."
- Next, select a target column for your experiment. In this case, let's choose 'Total Sales' as the target column.
- Excellent choice! Now, let's move on to the Data Preparation step. From the dropdown menu, select 'Preparation' and choose the 'SuperMarketPreparation'
- With the configuration complete, you'll be redirected to the 'Experiment Type' tab.
- Here, you can choose the prediction model based on your needs. Since we selected 'Total Sales' as the target column, let's choose the Regression option
- Perfect! Now that you've completed the necessary steps, click 'Done' to finalize your experiment setup.
- Congratulations! You have successfully created a new AutoML Regression experiment. You will receive a notification confirming its creation
- When you navigate to the AutoML tab, you will see a list of all your experiments. Your newly created experiment will appear at the top.
- Each experiment has a status associated with it, indicating its progress and outcome. Initially, a new experiment has the status of 'Started'
- As the model training progresses, the status changes to 'Running,' and you will receive notifications to keep you informed.

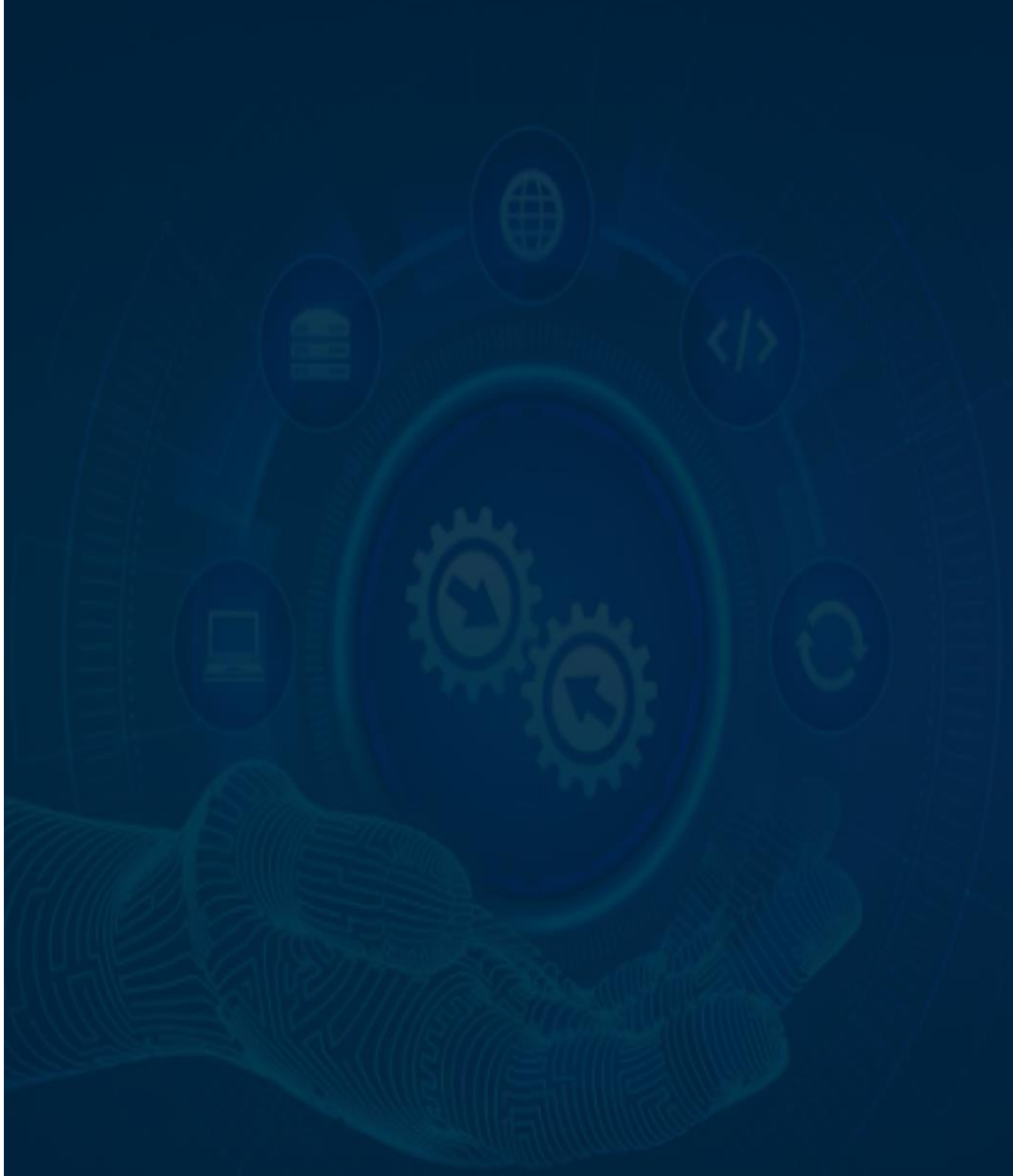


S.No.	Experiment	Description	Created	Updated	Status	Actions
1	Super Market Regression	Super Market Regression Exper...	Jul, 18, 2023 7:12 PM	Jul, 18, 2023 7:14 PM	Completed	View Report
2	Super Market Classification	Classification Super Market Ex...	Jul, 18, 2023 3:08 PM	Jul, 18, 2023 3:24 PM	Completed	View Report
3	SU2t		Jul, 17, 2023 7:10 PM	Jul, 17, 2023 7:11 PM	Completed	View Report
4	su2		Jul, 17, 2023 5:42 PM	Jul, 17, 2023 5:51 PM	Completed	View Report
5	S2		Jul, 17, 2023 5:38 PM	Jul, 17, 2023 5:38 PM	Failed	View Report
6	S1		Jul, 17, 2023 5:21 PM	Jul, 17, 2023 5:28 PM	Failed	View Report
7	workflow5_Regression		Jul, 17, 2023 5:14 PM	Jul, 17, 2023 5:22 PM	Completed	View Report
8	Workflow5		Jul, 17, 2023 5:08 PM	Jul, 17, 2023 5:18 PM	Failed	View Report

Items per page: 10 1 – 8 of 8

- Once the model training is successfully completed, the status changes to 'Completed,' indicated by a green colorOnce your AutoML experiment is completed, you have a range of options at your disposal to analyse the data and gain valuable insights. Utilizing tools such as Model Detail, Model Explainer, and Dataset Explainer etc By leveraging these options, you can gain deep insights into your data, interpret the models' behavior, and make informed decisions based on the analysis of your AutoML experiment.

Hope this Document has provided you with a clear understanding of how to leverage Automated Machine Learning for regression and classification tasks. By following this workflow, you can unlock valuable insights from your data, interpret the behavior of the models, and make informed decisions.



THANK YOU

For More Information,
Contact:

Email id: Sales@bdb.ai