

AUTO AI MODEL BUILDING IN IBM WATSON STUDIO

IBM-PBEL INTERNSHIP – PROJECT DOCUMENT

SUBMITTED BY – RISHIKA JALAN

BATCH – 2(ARTIFICIAL INTELLIGENCE)

COLLEGE – MODY UNIVERSITY SCIENCE AND TECHNOLOGY

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to IBM for providing me the opportunity to pursue my internship in the technology **Artificial Intelligence**. It has been an invaluable learning experience that has contributed immensely to my professional and personal development.

I am extremely grateful to my Mentor, **Mr. Samarth Amrute** for his invaluable guidance, encouragement and support during my time here. His insights and advice have pushed me to grow and learn in ways I didn't think possible. I could not have asked for a better mentor. I would also like to thank the batch mates. Thank you for making me feel so welcomed and for patiently answering all my questions. Your passion and dedication to your work is truly inspiring.

Finally, I want to thank my professors at my college **Mody University of Science and Technology** for laying the foundation that allowed me to maximize this internship opportunity. This internship has been a defining period in my life and I could not have done it without all of your help. Thank you.

INDEX

S.No.	TOPIC	Page No.
1.	INTRODUCTION	4
2.	TOOLS USED	5
3.	DATASET DESCRIPTION	6-9
4.	SETUP PROCESS	10-13
5.	MODEL BUILDING	14-17
6.	MODEL DEPLOYMENT	18-19
7.	MODEL TESTING	20-22
8.	CONCLUSION	23

INTRODUCTION

In this project, I used **IBM Watson Studio's AutoAI** to predict product sales across different Big Mart outlets. AutoAI automates the end-to-end machine learning process, including data preprocessing, feature engineering, model selection, and hyperparameter optimization — making it faster and easier to build high-quality predictive models.

The Big Mart Sales Prediction dataset includes historical sales data along with product details like item type, weight, price, and outlet characteristics such as size, location, and establishment year. By training an AutoAI model on this dataset, the goal is to accurately forecast sales, helping Big Mart make data-driven decisions to improve inventory management and marketing strategies.

This project showcases how automated machine learning can turn complex retail data into actionable insights, saving time and enhancing forecasting accuracy.

TOOLS USED

Services:

- IBM Watsonx.AI Studio
- IBM Watsonx.AI Runtime Service
- IBM Cloud Object Storage
- IBM Watsonx.AI AutoAI
- IBM Watsonx.AI Data Refinery

External Dataset:

Big Mart Sales Prediction

Source :

<https://www.kaggle.com/datasets/shivan118/big-mart-sales-prediction-datasets>

DATASET DESCRIPTION

1. Item_Identifier

- **Type:** Categorical (string)
 - **Description:**
This column contains a unique alphanumeric code assigned to each product (e.g., FDA15, DRC01). It acts as a primary identifier for items and is used to differentiate products, even those belonging to the same broader category. While the code itself does not directly carry numerical meaning, it can be helpful when merging data or creating derived features.
-

2. Item_Weight

- **Type:** Numerical (float)
 - **Description:**
Represents the weight of each product in kilograms. This attribute provides a physical characteristic of the product, which may influence logistics, pricing, and sales patterns. Some products have missing values in this column, which might need to be addressed through data cleaning (e.g., imputing missing weights using median or mean values by item type).
-

3. Item_Fat_Content

- **Type:** Categorical (string)
 - **Description:**
Indicates the fat content category of the product, such as "Low Fat" and "Regular." However, the data includes inconsistent labels like "low fat", "LF", and "reg" that refer to the same category. Proper preprocessing requires standardizing these labels to ensure meaningful analysis. Fat content can influence consumer preference and sales, especially for food and beverage items.
-

4. Item_Visibility

- **Type:** Numerical (float)
 - **Description:**
Represents the proportion of the total display area in the store that is allocated to a particular product. This is expressed as a percentage (typically as a decimal, e.g., 0.016). Higher visibility can positively impact product sales, as products displayed more prominently are likely to catch shoppers' attention. Extremely low or zero values might indicate missing or incorrect data.
-

5. Item_Type

- **Type:** Categorical (string)
 - **Description:**
Broad category or classification of products, such as Dairy, Soft Drinks, Meat, Fruits and Vegetables, Household, Baking Goods, etc. This column helps in grouping similar items and is useful for feature engineering. Typically, there are around 16–17 unique categories, offering a balanced view of the product range.
-

6. Item_MRP

- **Type:** Numerical (float)
 - **Description:**
Stands for Maximum Retail Price, which is the highest price at which the product is sold to consumers. This attribute is crucial, as pricing strongly affects consumer purchasing behavior and, consequently, sales volume. The distribution of MRP can reveal product pricing strategies and segmentation.
-

7. Outlet_Identifier

- **Type:** Categorical (string)
- **Description:**
Unique alphanumeric code for each retail outlet (e.g., OUT049, OUT018).

This serves to identify data coming from specific stores and is particularly important when analyzing outlet-level trends or building models that consider outlet characteristics.

8. Outlet_Establishment_Year

- **Type:** Numerical (integer)
 - **Description:**
The year in which the outlet was opened (e.g., 1999, 2009). Although this field is numeric, it often makes sense to derive a new feature from it, such as the **age of the outlet** (current year minus establishment year), which may provide more intuitive insights about how long the outlet has been operating and its potential effect on customer loyalty and sales.
-

9. Outlet_Size

- **Type:** Categorical (string)
 - **Description:**
Describes the physical size of the store, with categories like "Small", "Medium", and "High". Store size might reflect capacity, product variety, and expected customer traffic. Note that this column has missing values, which should be addressed through imputation or other data cleaning methods to avoid bias in analysis.
-

10. Outlet_Location_Type

- **Type:** Categorical (string)
 - **Description:**
Indicates the type of city where the outlet is located: "Tier 1", "Tier 2", or "Tier 3". Typically, Tier 1 cities are larger metropolitan areas, while Tier 3 cities are smaller towns. This column can capture differences in customer behavior, purchasing power, and competition across different urban settings.
-

11. Outlet_Type

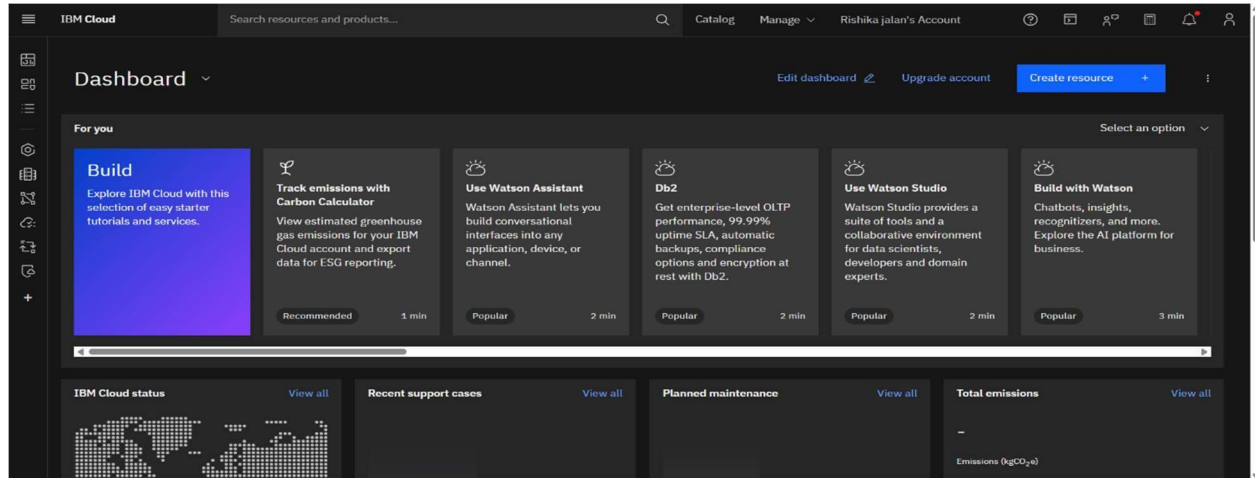
- **Type:** Categorical (string)
 - **Description:**
Describes the format of the retail outlet, such as "Supermarket Type1", "Supermarket Type2", "Supermarket Type3", or "Grocery Store". These categories represent differences in business models, product variety, and scale of operations, which can significantly impact sales performance.
-

12. Item_Outlet_Sales

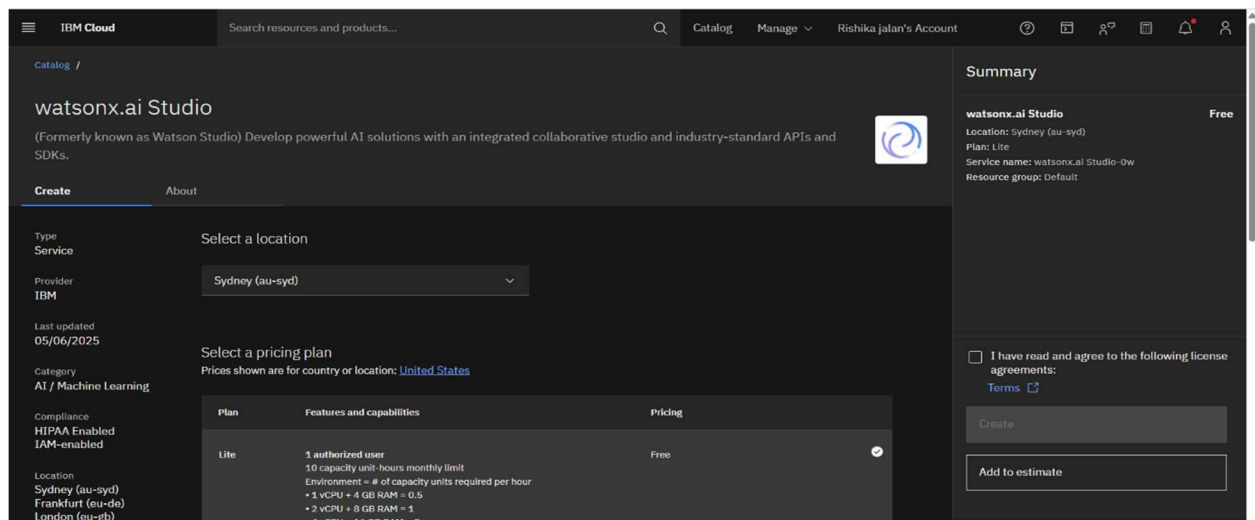
- **Type:** Numerical (float)
- **Description:**
The target variable of the dataset, representing the total sales of each product in a particular outlet over a specified period (often monthly or quarterly). This column is measured in monetary units and is used as the dependent variable for regression models aiming to predict future sales.

SETUP PROCESS

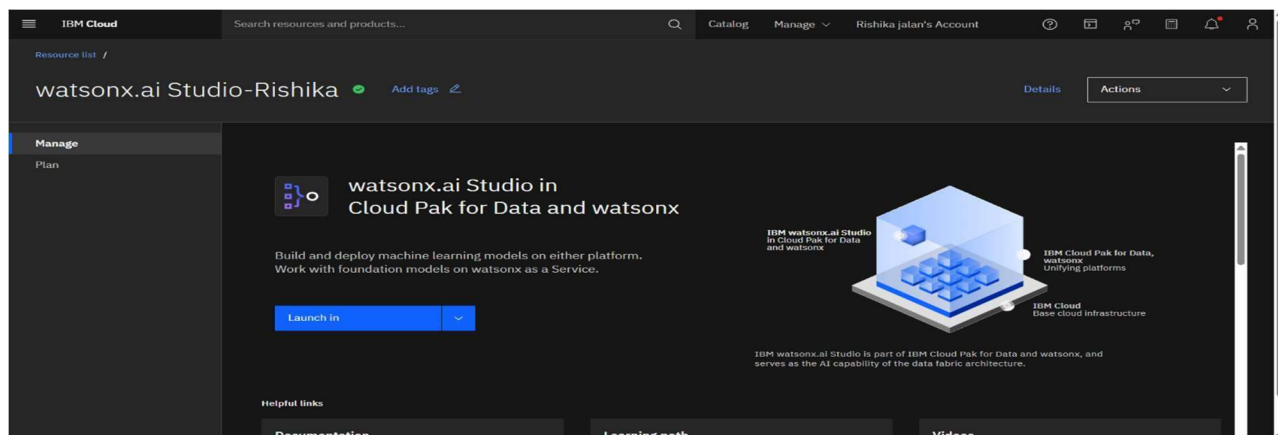
STEP 1: Login to IBM Cloud account



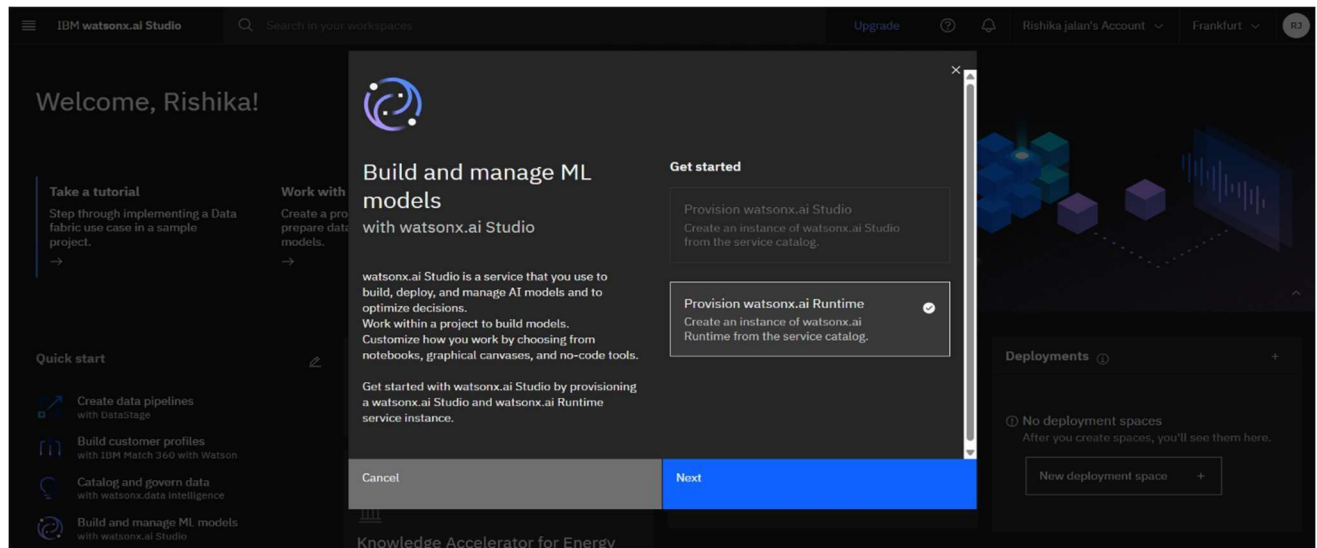
STEP 2: Creating Watsonx.ai Studio Service Instance



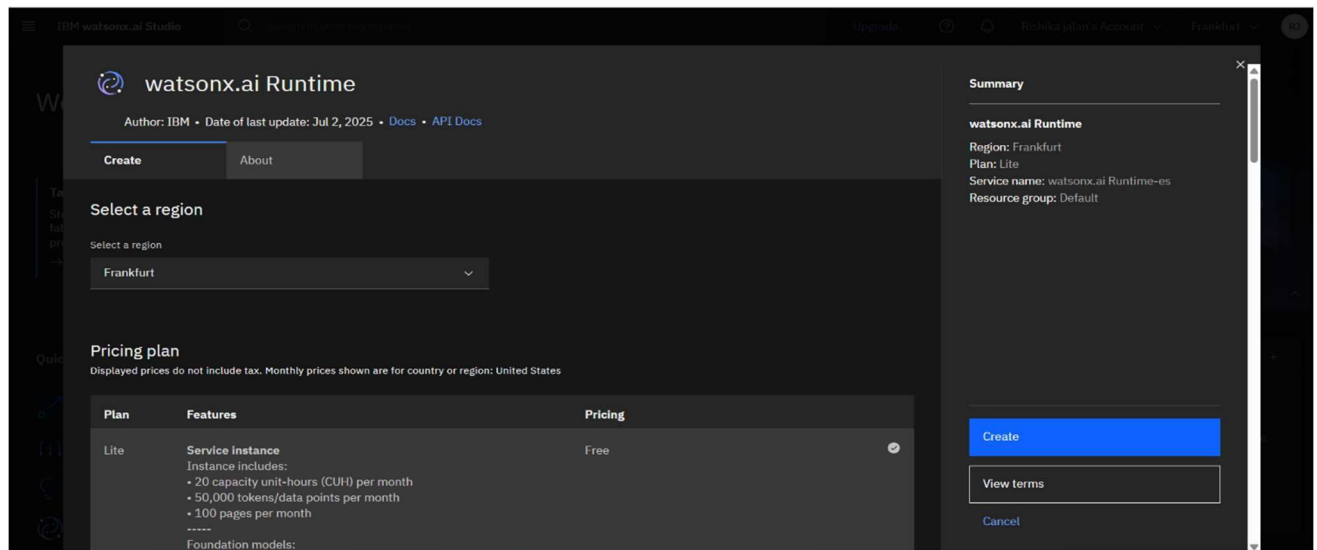
STEP 3: Launching Watsonx.ai Studio – Cloud Pak for Data and Watson



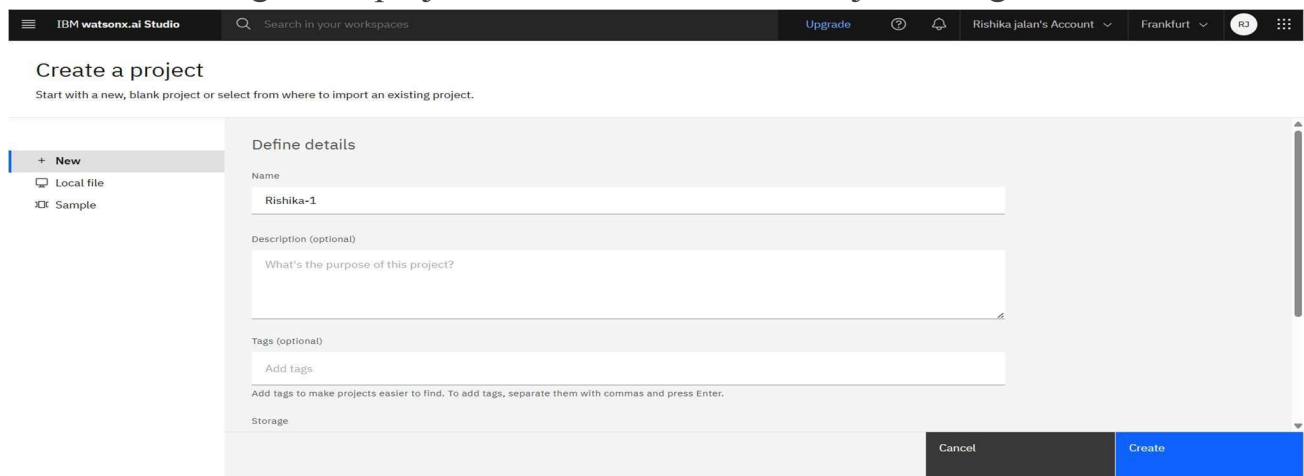
STEP 4: Watsonx.ai Studio Launched



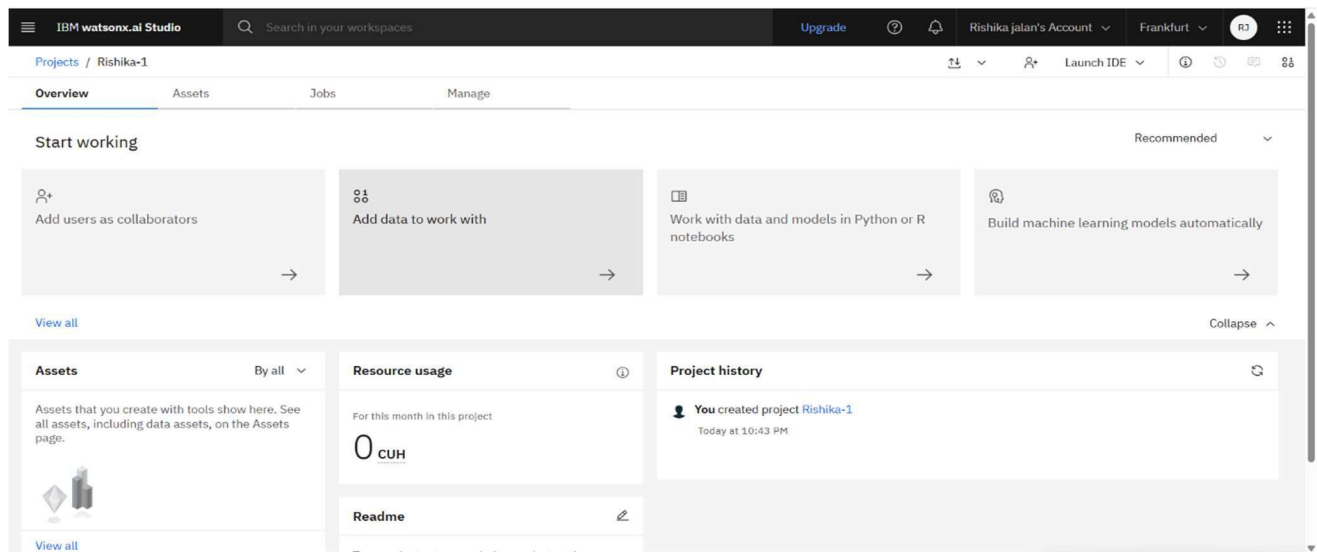
STEP 5: Creating Watsonx.ai Studio Runtime service instance



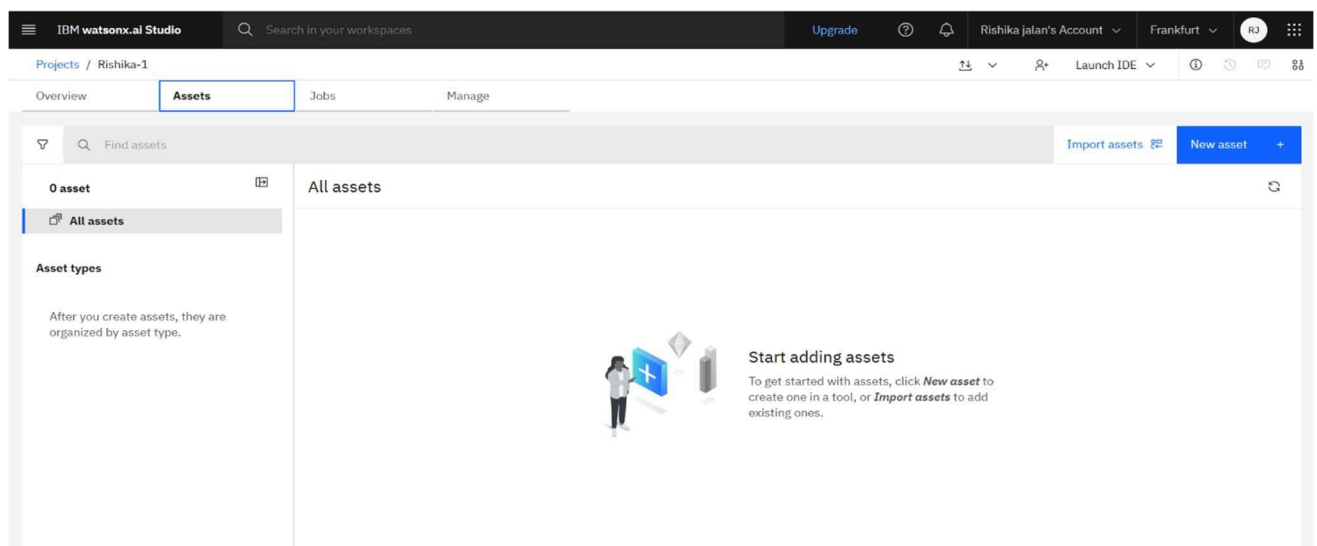
STEP 6: Creating a new project with associated cloud object storage



STEP 7: Overview of the watsonx.ai studio project

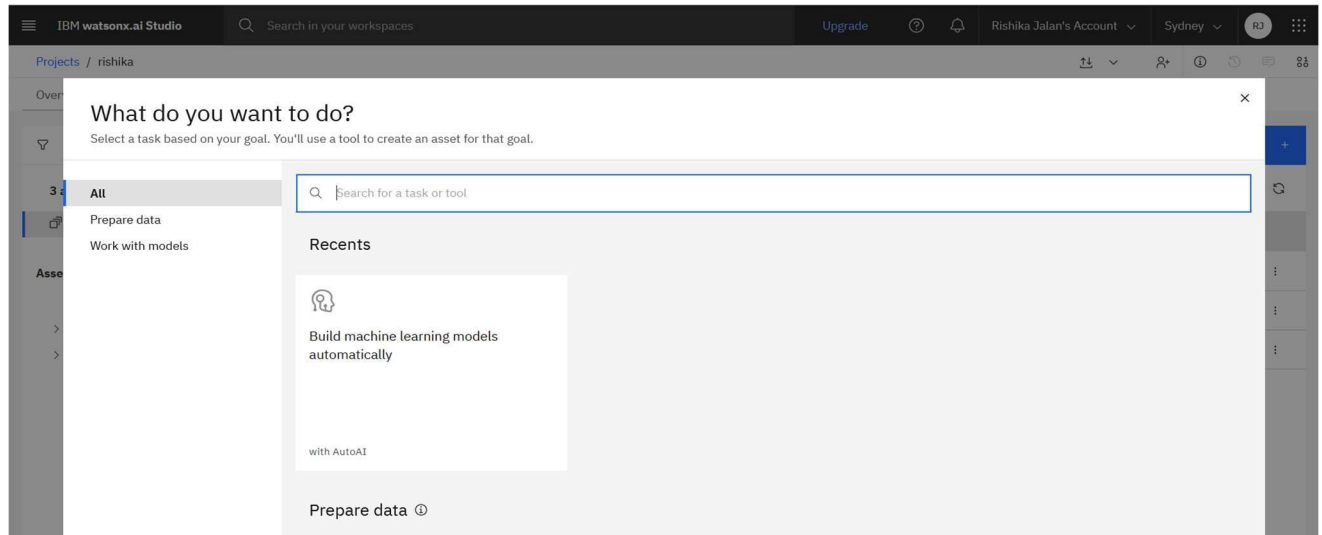


STEP 8: Assets of watsonx.ai Studio project

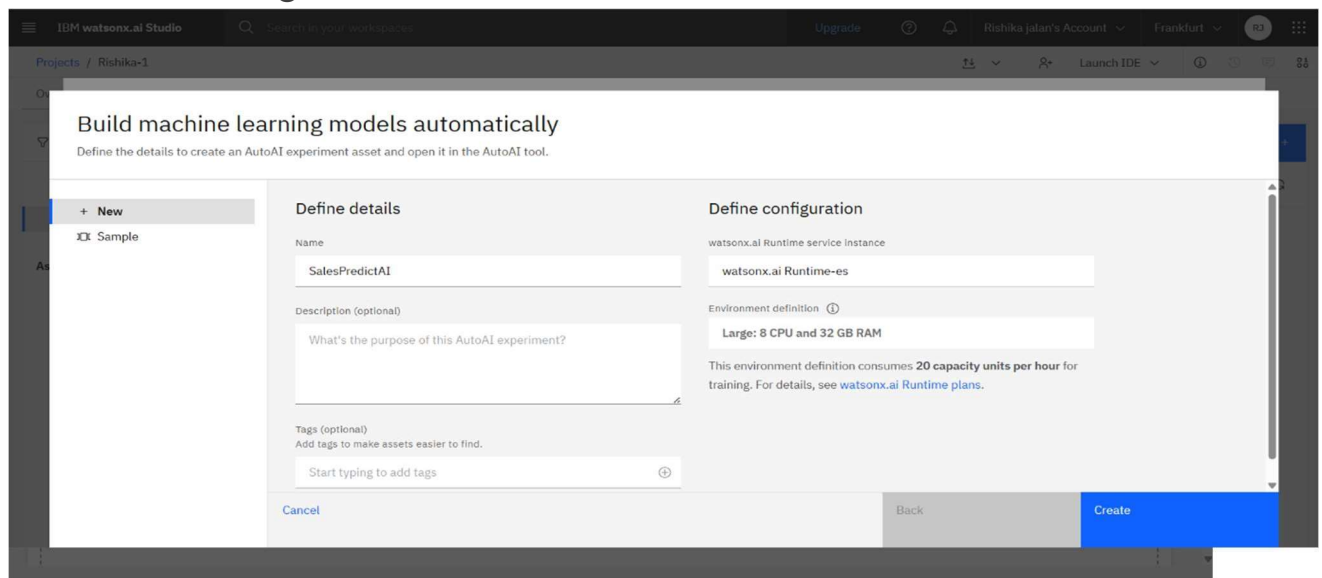


MODEL CREATION USING AUTOAI

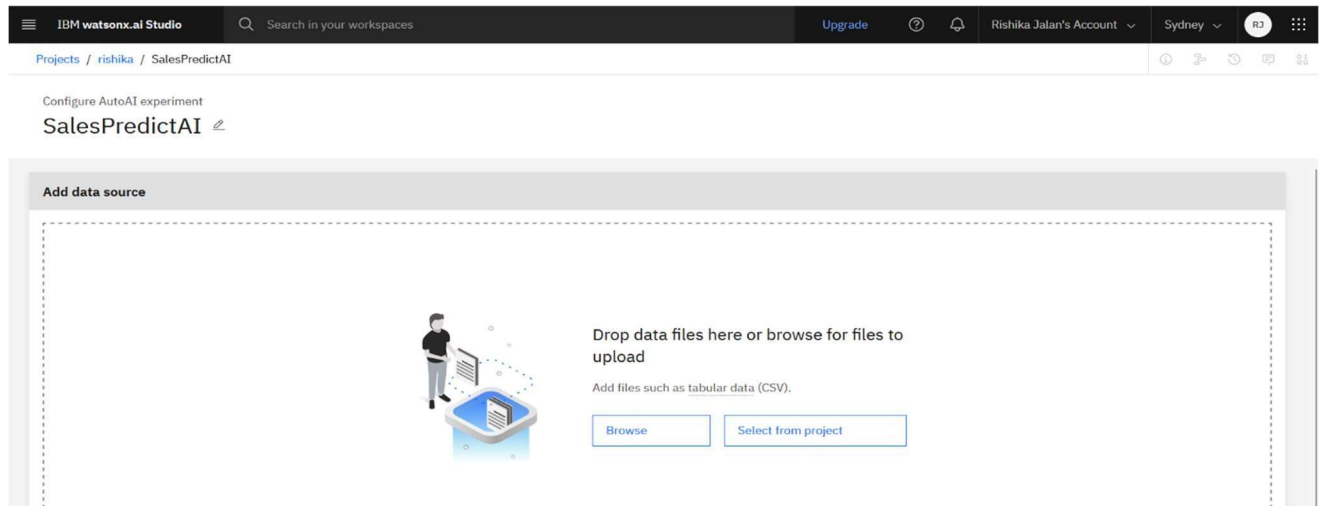
STEP 1: Selecting new asset as build machine learning model automatically



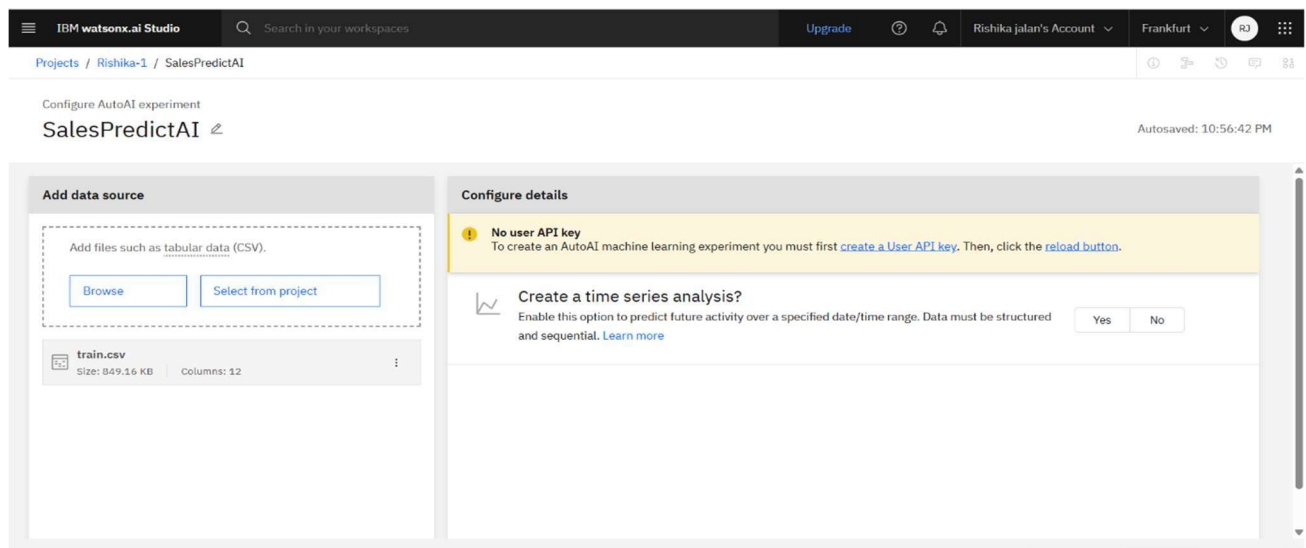
STEP 2: Connecting watsonx.ai runtime service instance



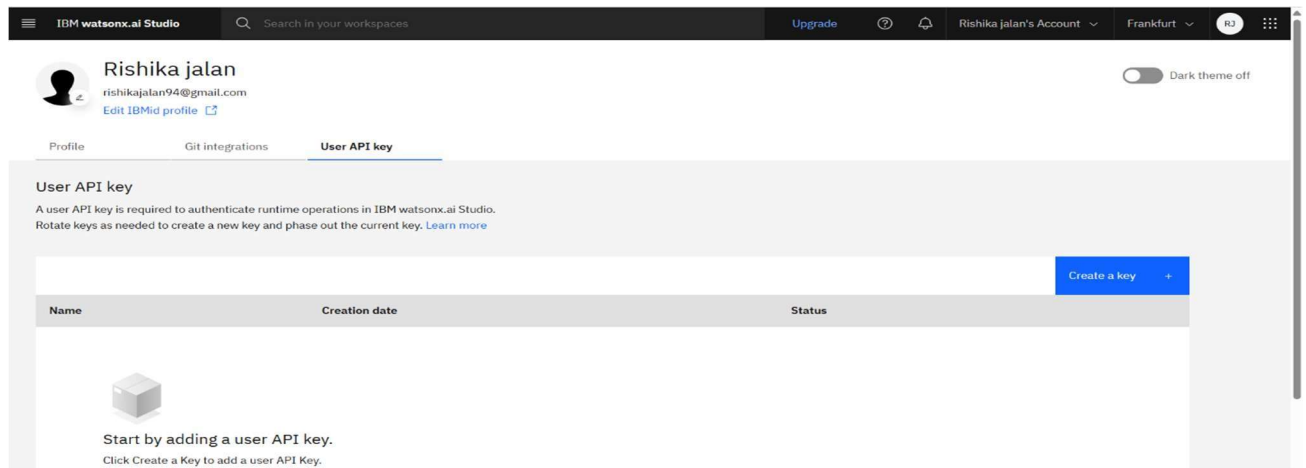
STEP 3: Importing Dataset from local files though Browse



STEP 4: Configure details after fetching training dataset



STEP 5: Creating User API Key



STEP 6: Selecting features to be predicted with type

The screenshot shows the IBM Watson AI Studio interface. At the top, there's a navigation bar with 'IBM watsonx.ai Studio', a search bar, and user account information. Below the navigation bar, the breadcrumb trail shows 'Projects / Rishika-1 / SalesPredictAI'. The main area is titled 'Configure AutoAI experiment SalesPredictAI'. On the left, there's a file upload section with a 'Browse' button and a 'Select from project' button. Below this, a file named 'train.csv' is listed with a size of 849.16 KB and 12 columns. On the right, there's a configuration panel. It has a toggle to 'Enable this option to predict future activity over a specified date/time range. Data must be structured and sequential.' with 'Yes' and 'No' buttons. Below this, a section titled 'What do you want to predict?' shows 'Item_Outlet_Sales' as the prediction column. The prediction type is set to 'Regression', and it's optimized for 'RMSE & run time'. At the bottom right, there's a 'Run experiment' button. The status 'CUH remaining: 20 CUH' is also visible.

STEP 7: Reading training data in train.csv

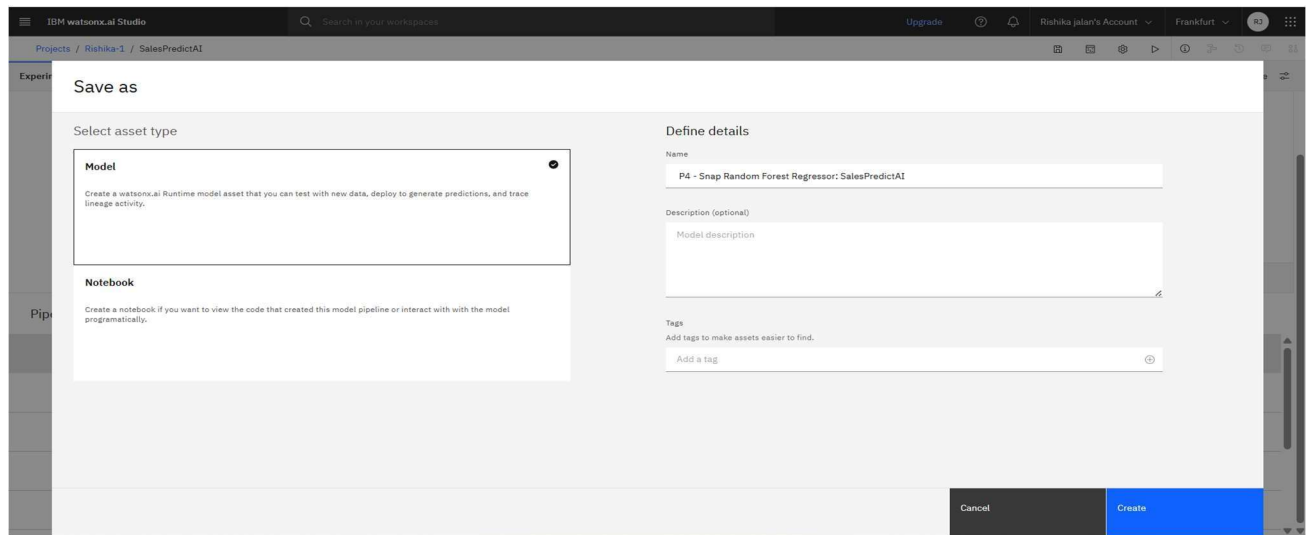
The screenshot shows the IBM Watson AI Studio interface. The breadcrumb trail is 'Projects / Rishika-1 / SalesPredictAI'. The main area is titled 'Experiment summary'. Below this, there's a 'Relationship map' section showing a diagram of the data flow. The diagram shows 'train.csv' being split into '90% TRAINING DATA (3 folds)' and '10% HOLDOUT DATA'. To the right, there's a 'Progress map' section showing the progress of the experiment. It includes a 'Swap view' button and a 'View log' button. The status 'Experiment completed' is visible, along with '8 PIPELINES GENERATED' and 'Time elapsed: 4 minutes'.

STEP 8: Building pipeline automatically with AutoAI

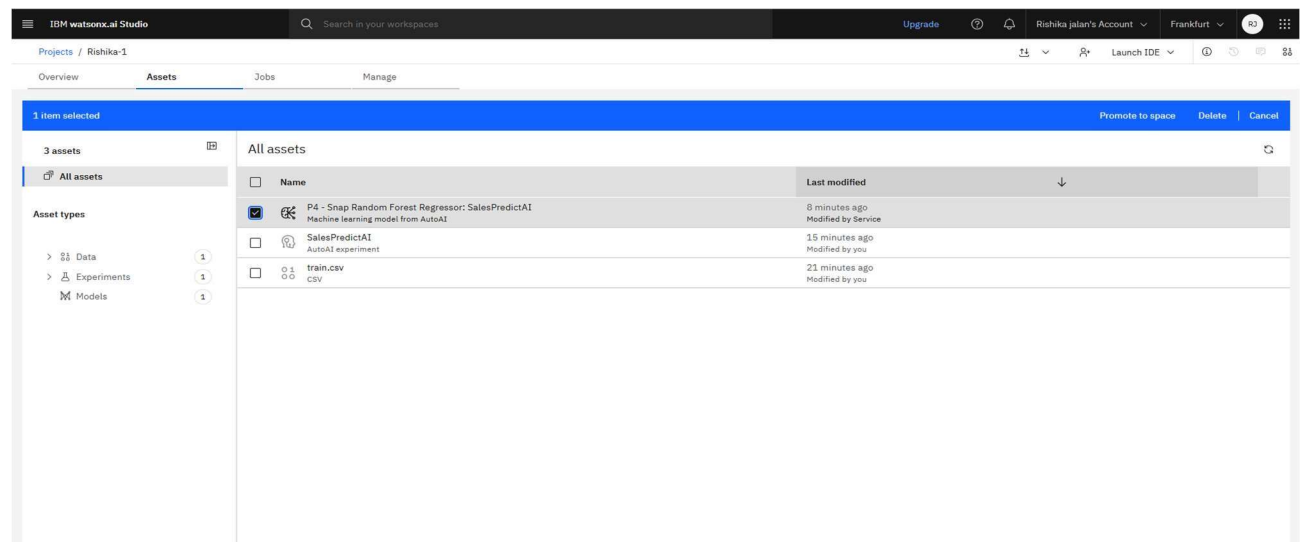
The screenshot shows the IBM Watson AI Studio interface. The breadcrumb trail is 'Projects / Rishika-1 / SalesPredictAI'. The main area is titled 'Experiment summary'. Below this, there's a 'Pipeline leaderboard' section showing a table of generated pipelines. The table has columns for Rank, Name, Algorithm, RMSE (Optimized) Cross Validation, Enhancements, and Build time. The top pipeline is 'Pipeline 4' using 'Snap Random Forest Regressor' with an RMSE of 1067.382. To the right, there's a 'Progress map' section showing the progress of the experiment. It includes a 'Swap view' button and a 'View log' button. The status 'Experiment completed' is visible, along with '8 PIPELINES GENERATED' and 'Time elapsed: 4 minutes'.

Rank	Name	Algorithm	RMSE (Optimized) Cross Validation	Enhancements	Build time
1	Pipeline 4	Snap Random Forest Regressor	1067.382	HPD-1 FE HPD-2	00:00:46
2	Pipeline 3	Snap Random Forest Regressor	1067.504	HPD-1 FE	00:00:19

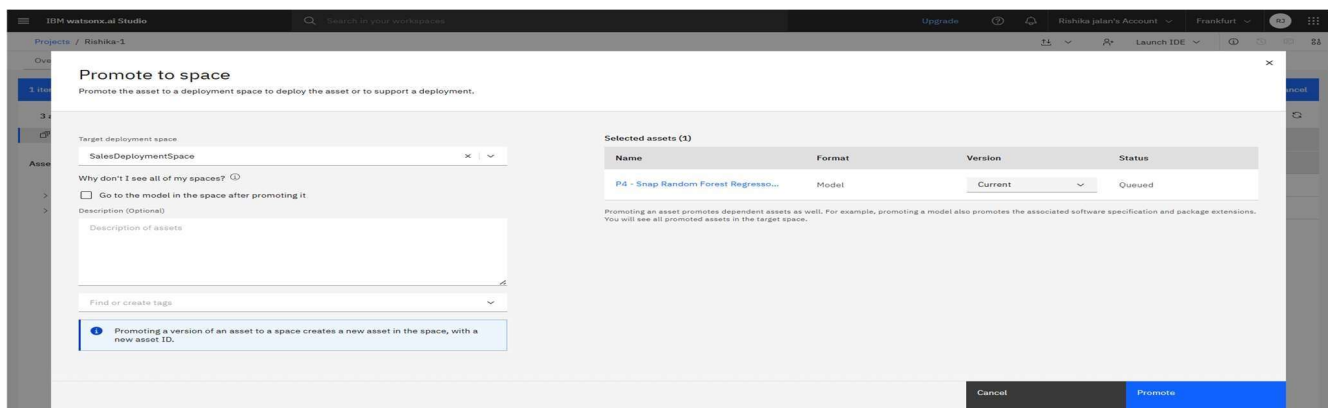
STEP 9: Saving the best Model according to Pipeline Leaderboard – Snap Random Forest Regression



STEP 10: Promote to space



STEP 11: Promoting to space using deployment space



STEP 12: Promoted to space

Promote to space
Promote the asset to a deployment space to deploy the asset or to support a deployment.

✔ Promotion completed.

Selected assets (1)

Name	Format	Version	Status
P4 - Snap Random Forest Regressor: SalesPredictAI	Model	Current	Promoted ✔

Promoting an asset promotes dependent assets as well. For example, promoting a model also promotes the associated software specification and package extensions. You will see all promoted assets in the target space.

Close

Success
Successfully promoted P4 - Snap Random Forest Regressor: SalesPredictAI to the deployment space. Go to the [deployment space](#) to prepare the assets for deployment.
Timestamp 11:28:06 PM

MODEL DEPLOYMENT

STEP 1: Creating a Deployment Space

The screenshot shows the 'Create a deployment space' dialog box in IBM watsonx.ai Studio. The dialog has a title bar 'Create a deployment space' and a subtitle 'Use a space to collect assets in one place to create, run, and manage deployments'. It contains a 'Define details' section with the following fields:

- Name:** SalesDeploymentSpace
- Description (Optional):** What's the purpose of this space? (0/100 characters)
- Deployment stage:** Select or enter a name that describes the purpose of the space (dropdown menu)
- Tags (optional):** Find or create tags (dropdown menu)

At the bottom right of the dialog are 'Cancel' and 'Create' buttons.

STEP 2: Deployment Space – SalesDeploymentSpace

The screenshot shows the 'Deployments' page in IBM watsonx.ai Studio. The page has a header 'Deployments' with a subtitle '1 space' and a 'New deployment space' button. Below the header is a table with the following columns: Name, Last modified, Your role, Collaborators, Tags, Type, Online deployments, and Jobs. The table contains one row for 'SalesDeploymentSpace'.

Name	Last modified	Your role	Collaborators	Tags	Type	Online deployments	Jobs
SalesDeploymentSpace	Jul 9, 2025, 11:15 PM	Admin				0	0

At the bottom of the table, there is a pagination bar showing 'Items per page: 20' and '1-1 of 1 items'.

STEP 3: Connecting watsonx.ai runtime service

The screenshot shows the 'SalesDeploymentSpace' details page in IBM watsonx.ai Studio. The page has a header 'SalesDeploymentSpace' and a subtitle 'Overview'. The page is divided into two main sections: 'General' and 'Controls'.

General section:

- Name:** SalesDeploymentSpace
- Description:** No description provided.
- Space GUID:** 860c2c76-e79d-43c2-9f19-c19d0ce42a2d
- Date created:** Jul 9, 2025, 11:15 PM by Rishika jalan (You)
- Stage:** Not provided
- Stage type:** Pre-production
- Tags:** No tags are set to this space.

Controls section:

- Cloud Pak for Data platform:** Switch platform
- Grant access:** Grant access

Storage used section:

- Storage used:** 72.27 KB
- Name:** Cloud Object Storage-qw
- Bucket:** f64bad32-4b04-4a78-a320-2172bd39c9ec
- Manage in IBM Cloud:** Manage in IBM Cloud
- watsonx.ai Runtime service:** watsonx.ai Runtime-es

STEP 4: New deployment

The screenshot shows the IBM Watsonx AI Studio interface. The top navigation bar includes the IBM Watsonx AI Studio logo, a search bar, and user account information. The main content area is titled 'Deployments' and shows a table with columns: Name, Type, Status, Tags, and Last modified. The table is currently empty, and a message states: 'This asset doesn't have any deployments yet. Use the New Deployment button to create a deployment for this asset.' A 'New deployment' button is visible in the top right corner of the table area. On the right side, there is a sidebar titled 'About this asset' which provides details about the asset, including its name, description, and asset details.

Name	Type	Status	Tags	Last modified
------	------	--------	------	---------------

STEP 5: Creating an online deployment

The screenshot shows the 'Create a deployment' dialog in IBM Watsonx AI Studio. The dialog has two tabs: 'Online' and 'Batch'. The 'Online' tab is selected, showing options to run the model on data in real-time. The 'Batch' tab is also visible, showing options to run the model against data as a batch process. The dialog includes fields for Name, Serving name, Description, and Tags. The 'Name' field is filled with 'SalesDeployment1'. The 'Serving name' field is filled with 'Deployment serving name'. The 'Description' field is filled with 'Deployment description'. The 'Tags' field is filled with 'Find or create tags'. The dialog has 'Cancel' and 'Create' buttons at the bottom right.

STEP 6: Deployed model by creating new Online deployment

The screenshot shows the IBM Watsonx AI Studio interface after creating a new online deployment. The 'Deployments' table now contains one entry: 'SalesDeployment1'. The table has columns: Name, Type, Status, Tags, and Last modified. The 'Status' column shows 'Deployed' with a green checkmark. The 'Last modified' column shows '8 minutes ago' by 'Rishika jalan (You)'. The 'New deployment' button is still visible in the top right corner of the table area. On the right side, the 'About this asset' sidebar is still present, showing details about the asset.

Name	Type	Status	Tags	Last modified
SalesDeployment1	Online	Deployed		8 minutes ago Rishika jalan (You)

MODEL TESTING

STEP 1: SalesDeployment1 API reference for testing

The screenshot displays the IBM Watsonx AI Studio interface. The top navigation bar includes the IBM Watsonx AI Studio logo, a search bar, and user account information (Rishika Jalan's Account, Frankfurt, 83). The main content area is titled "SalesDeployment1" and shows it is "Deployed" and "Online". The "API reference" tab is selected, showing endpoints for scoring. The "Private endpoint" is `https://private.eu-de.ml.cloud.ibm.com/ml/v4/deployments/fc44873e-145b-48a6-bbea-a8aba9932cc0/predictions?version=2021-05-01` and the "Public endpoint" is `https://eu-de.ml.cloud.ibm.com/ml/v4/deployments/fc44873e-145b-48a6-bbea-a8aba9932cc0/predictions?version=2021-05-01`. The "Code snippets" section shows a cURL command for testing the API. The right sidebar provides details about the deployment, including its name, description, deployment ID, serving name, software specification, and associated asset.

Endpoints for scoring

Private endpoint: `https://private.eu-de.ml.cloud.ibm.com/ml/v4/deployments/fc44873e-145b-48a6-bbea-a8aba9932cc0/predictions?version=2021-05-01`

Public endpoint: `https://eu-de.ml.cloud.ibm.com/ml/v4/deployments/fc44873e-145b-48a6-bbea-a8aba9932cc0/predictions?version=2021-05-01`

Code snippets

cURL

```
# NOTE: you must set $API_KEY below using information retrieved from your IBM Cloud account (https://eu-de.dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/ml-authentication.html#)
export API_KEY=your API key

export IAM_TOKEN=$(curl --insecure -X POST --location "https://iam.cloud.ibm.com/identity/token" \
--header "Content-Type: application/x-www-form-urlencoded" \
--header "Accept: application/json" \
--data-urlencode "grant_type=urn:ibm:params:oauth:grant-type:apikey" \
--data-urlencode "apikey=$API_KEY" | jq -r .access_token)

# TODO: manually define and pass values to be scored below
curl --location "https://private.eu-de.ml.cloud.ibm.com/ml/v4/deployments/fc44873e-145b-48a6-bbea-a8aba9932cc0/predictions?version=2021-05-01" \
--header "Content-Type: application/json"
```

About this deployment

Name: SalesDeployment1

Description: No description provided.

Deployment Details

Deployment ID: fc44873e-145b-48a6-bbea-a8aba9932cc0

Serving name: No serving name.

Software specification: hybrid_0.1

Hybrid pipeline software specifications: autoai-kb_r24.1-py3.11

Copies: 1

Tags: Add tags to make assets easier to find.

Associated asset: P4 - Snap Random Forest Regressor: SalesPredictAI

Last modified: 9 minutes ago

Created on: Jul 9, 2025

STEP 2: Importing Testing Dataset from local file to test the Model

The screenshot displays the IBM Watsonx AI Studio interface. The top navigation bar includes the IBM Watsonx AI Studio logo, a search bar, and user account information (Rishika Jalan's Account, Frankfurt, 83). The main content area is titled "SalesDeployment1" and shows it is "Deployed" and "Online". The "Test" tab is selected, showing the "Enter input data" section. The "Text" input type is selected, and the "JSON" format is chosen. The "Enter data manually or use a CSV file to populate the spreadsheet. Max file size is 50 MB." section is visible. The "Input data" section shows a CSV file named "autoai_kb_input_schema" being uploaded. The "Download CSV template" and "Browse local files" buttons are present. The "Search in space" button is also visible. The "Clear all" button is located in the top right corner of the input area. The table below shows the input data for the model.

Enter input data

Text

JSON

Enter data manually or use a CSV file to populate the spreadsheet. Max file size is 50 MB.

Input data

autoai_kb_input_schema

Download CSV template

Browse local files

Search in space

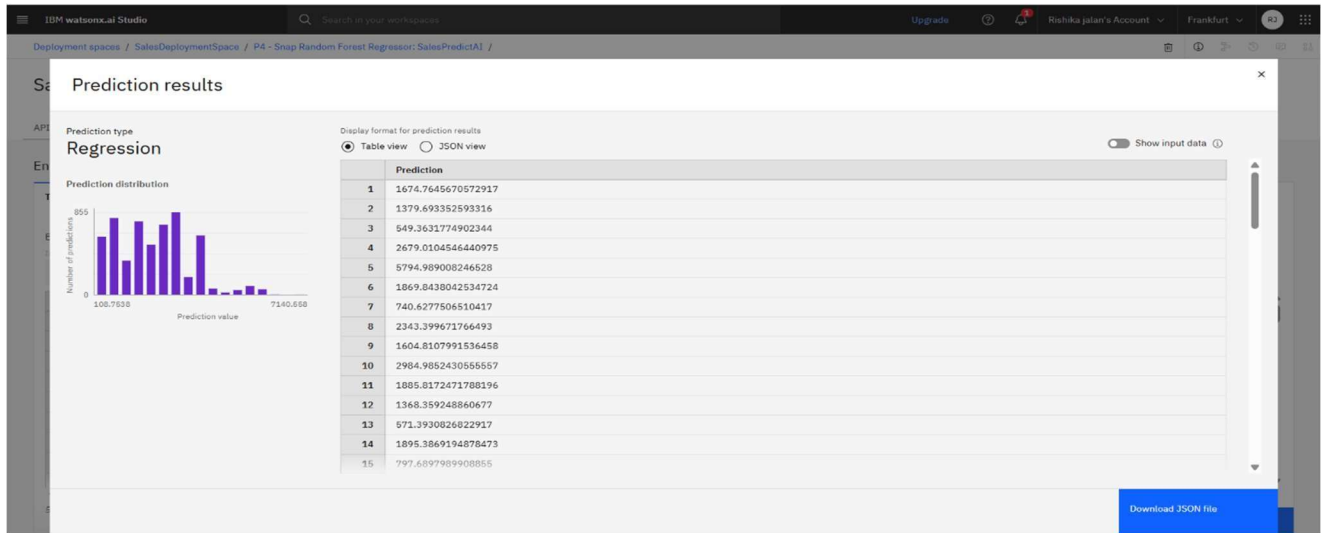
Clear all

	Item_Identifier (other)	Item_Weight (double)	Item_Fat_Content (other)	Item_Visibility (double)	Item_Type (other)	Item_MRP (double)	Outlet_Identifier (other)	Outlet_Establishment_Year (double)	Outlet_Size (other)
1	FDW58	20.75	Low Fat	0.007564836	Snack Foods	107.8622	OUT049	1999	Medium
2	FDW14	8.3	reg	0.038427677	Dairy	87.3198	OUT017	2007	Medium
3	NCN55	14.6	Low Fat	0.099574908	Others	241.7538	OUT010	1998	Medium
4	FDQ58	7.315	Low Fat	0.015388393	Snack Foods	155.034	OUT017	2007	Medium
5	FDY38		Regular	0.118599314	Dairy	234.23	OUT027	1985	Medium
6	FDH56	9.8	Regular	0.063817206	Fruits and Vegetab	117.1492	OUT046	1997	Small
7	FDL48	19.35	Regular	0.082601537	Baking Goods	50.1034	OUT018	2009	Medium
8	FDC48		Low Fat	0.015782495	Baking Goods	81.0592	OUT027	1985	Medium

5,682 rows, 11 columns

Predict

STEP 3: Predicted Result of the test dataset



STEP 4: Create API Keys

API keys

Create, view, and work with API keys that you have access to manage. IBM Cloud API keys are associated with a user's identity and can be used to access cloud platform and classic infrastructure APIs, depending on the access that is assigned to the user. The following table displays a list of API keys created in this account. [Learn more.](#)

Looking for more options to manage API Keys? Try [IBM Cloud® Secrets Manager](#) for creating and leasing API keys dynamically and storing them securely in your own dedicated instance.

Unused or overly permissive API keys increase the risk of unauthorized access. Regularly review the [Inactive identities report](#), rotate keys, and apply only the minimum required permissions.

API keys associated with a user's identity have the same access that the user is assigned across all accounts. To update the access for an API key, assign or remove access for the user.

View: My IBM Cloud API keys

Filter by API key name or description

Status	Name	Description	Date created	Enabled
🔒	cpd-apikey-IBMId-6930010692-2025-07-09T17:20:40Z	API key created/managed by task credentials. It is managed for your use with Watson Studio operations. Please do not delete here.	7-9-2025 17:20 GMT	Yes
🔒	SalesKey		7-9-2025 18:35 GMT	Yes

STEP 5: API Key created successfully

API key successfully created

Copy the API key or click download to save it. You won't be able to see this API key again, so you can't retrieve it later. The API key is no longer displayed after 252 seconds.

API key:

Copy Download

Status	Name	Description	Date created	Enabled
🔒	cpd-apikey-IBMId-6930010692-2025-07-09T17:20:40Z	API key created/managed by task credentials. It is managed for your use with Watson Studio operations. Please do not delete here.	7-9-2025 17:20 GMT	Yes
🔒	SalesKey		7-9-2025 18:35 GMT	Yes

Items per page: 25 1-25 items Page 1

STEP 6: Using API key and public endpoints to predict the results for some sample data on Google Colab Notebook

```
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account (https://eu-de.dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/ml-authentic)
API_KEY = "jz8vstPj_b0s2kw_B0wPigJf0t27jnaCgYwiv_eZYvhj"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'content-type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [
    {
        "fields": ["Item_Identifier",
                    "Item_Weight",
                    "Item_Fat_Content",
                    "Item_Visibility",
                    "Item_Type",
                    "Item_MRP",
                    "Outlet_Identifier",
                    "Outlet_Establishment_Year",
                    "Outlet_Size",
                    "Outlet_Location_Type",
                    "Outlet_Type"],
        "values": [
            ["FDA15", 9.3, "Low Fat", 0.016047, "Dairy", 249.8092, "OUT049", 1999, "Medium", "Tier 1", "Supermarket Type1"],
            ["DRC01", 5.92, "Regular", 0.019278, "Soft Drinks", 48.2692, "OUT018", 2009, "Medium", "Tier 3", "Supermarket Type2"],
            ["FDN15", 17.5, "Low Fat", 0.016760, "Meat", 141.6180, "OUT049", 1999, "Medium", "Tier 1", "Supermarket Type1"],
            ["FDX07", 19.2, "Regular", 0.000000, "Household", 182.0950, "OUT010", 1998, "Small", "Tier 3", "Grocery Store"],
            ["MCD19", 8.93, "Low Fat", 0.000000, "Baking Goods", 53.8614, "OUT013", 1987, "High", "Tier 3", "Supermarket Type1"],
            ["FDP36", 10.395, "Regular", 0.124952, "Snack Foods", 51.4008, "OUT027", 1985, "Medium", "Tier 3", "Supermarket Type3"],
            ["FPO10", 13.65, "Low Fat", 0.149963, "Frozen Foods", 57.6588, "OUT045", 2002, "Small", "Tier 2", "Supermarket Type1"],
            ["FDH17", 16.2, "Low Fat", 0.054447, "Dairy", 107.7622, "OUT017", 2007, "Medium", "Tier 2", "Supermarket Type2"],
            ["FDG33", 15.3, "Regular", 0.046845, "Canned", 162.4160, "OUT035", 2004, "Small", "Tier 1", "Supermarket Type1"],
            ["FDM33", 18.5, "Low Fat", 0.035685, "Breads", 163.5254, "OUT046", 1997, "High", "Tier 2", "Supermarket Type1"],
        ]
    }
]}

response_scoring = requests.post('https://eu-de.ml.cloud.ibm.com/ml/v4/deployments/fc44073e-145b-48a6-bbea-a8aba9932cc0/predictions?version=2021-05-01', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})

print("Scoring response")
try:
    print(response_scoring.json())
except ValueError:
    print(response_scoring.text)
except Exception as e:
    print(f"An unexpected error occurred: {e}")

Scoring response
{'predictions': [{'fields': ['prediction'], 'values': [[3910.720648871528], [735.0598836263022], [1944.67979730903], [471.5779534233941], [842.7922526041667], [1497.1166883680555], [8
```

CONCLUSION

In this project, I leveraged **IBM Watson Studio's AutoAI** to develop a predictive model for forecasting product sales across Big Mart outlets. AutoAI simplified the machine learning workflow by automating key steps such as data preprocessing, feature engineering, and model optimization, which significantly reduced development time and complexity.

By training on the **Big Mart Sales Prediction dataset**, the model provided useful insights into how product attributes and outlet characteristics affect sales. These predictions can help Big Mart improve inventory planning, optimize stock levels, and make data-driven marketing decisions.

Overall, this project highlights the power of **AutoAI** in turning complex retail data into actionable business strategies. Moving forward, further improvements could include integrating more recent sales data, exploring additional features, and deploying the model as an API to support real-time decision-making.