A

REPORT

ON

Electricity Generation Analysis using BDB tool

BY

DARREN ANDREW DCUNHA 20191COM0045 COMPUTER ENGINEERING

Prepared in partial fulfillment of the

PIP 103, Professional Practice – 3

AT

BIZVIZ TECHNOLOGIES
Bangalore

PRESIDENCY UNIVERSITY,

BENGALURU JUNE, 2023

A

REPORT

ON

Electricity Generation Analysis using BDB tool

BY

DARREN ANDREW DCUNHA 20191COM0045 COMPUTER ENGINEERING

Prepared in partial fulfillment of the

PIP 103, Professional Practice – 3

AT

BIZVIZ TECHNOLOGIES
Bangalore

PRESIDENCY UNIVERSITY,

BENGALURU JUNE, 2023

DEPARTMENT OF COMPUTER ENGINEERING
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING & INFORMATION SCIENCE (SOCSE & IS)

PRESIDENCY UNIVERSITY

CERTIFICATE

This is to certify that the Project report **"Electricity Generation Analysis using BDB tool"** being submitted by **"DARREN ANDREW DCUNHA"** bearing roll number(s): 20191COM0045, in partial fulfillment of requirement for the award of degree of **Bachelor of Technology** in **Computer Engineering** is a bonafide work carried out under my supervision.

Dr. Mohammed Mujeer Ulla Assistant Professor SOCSE & IS PresidencyUniversity	**Mr. Prakash B Metre** Internship Coordinator & Asst. Professor- SOCSE & IS Presidency University	**Dr. Kalaiarasan C.** Associate Dean- SOCSE & IS Presidency University	**Dr. Md Sameeruddin Khan** Dean- SOCSE & IS Presidency University

DEPARTMENT OF COMPUTER ENGINEERING
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING & INFORMATION SCIENCE(SOCSE & IS)
PRESIDENCY UNIVERSITY

DECLARATION

I hereby declare that the report entitled "Electricity Generation Analysis using BDB tool" which is being submitted to the Presidency University, Bangalore is a bonafide report of the work carried out by me. The material contained in this report has not been submitted to any University or Institution for the award of any degree.

Name:	ID number:	Department:
DARREN ANDREW DCUNHA	20191COM0045	Computer Engineering

ACKNOWLEDGEMENT

I am thankful to my institute, Presidency University and the Professional Practice Department for structuring the course of Professional Practice-III, thus allowing me to do a professional and technical project.

I am also thankful to **"BIZVIZ TECHNOLOGIES"** for training me in the **"End-to-End Business Analytics"** all the necessary inputs for training

I would like to express my gratitude to **"BIZVIZ TECHNOLOGIES, Karthikeyan P.S The Solution Architect"** for his precious contributions in bringing this training to life and sharing his ideas.

I express my sincere thanks to our respected dean **Dr. Md Sameeruddin Khan**, Dean-CSE & IS, School of Engineering, Presidency University for getting me permission to undergo the Internship.

I record my heartfelt gratitude to our beloved professors **Dr. C. Kalaiarasan,** Associate Dean-CSE and **Dr. T K Thivakaran**, Prof. & HOD-In-charge for rendering timely help for the successful completion of this Professional Practice (Internship) work.

I take this opportunity to thank my guide **Dr.Mohammed Mujeer Ulla** for his inspirational guidance, valuable suggestions and providing me a chance for the completion of the Professional Practice work.

I am greatly indebted to our Internship coordinator **Mr. Prakash B Metre,** Assistant Professor, Department of Computer Science and Engineering, Presidency University for his constant support, guidance and encouragement and all the members of Internship team.

I thank my friends for the strong support and inspiration they have provided me in bringing out this Internship.

DARREN ANDREW DCUNHA

PRESIDENCY UNIVERSITY, BENGALURU

Professional Practice Program
Professional Practice (IP) – II

PP Centre: **BIZVIZ TECHNOLOGIES**

Start Date: 16/01/2023 **End Date:** 17/04/2023

Title of the Project: Electricity Generation Analysis using BDB tool

Name of the Student	ID No.	Branch
Darren Andrew Dcunha	20191COM0045	COMPUTER ENGINEERING

Name of the Expert	Designation	Department
Karthikeyan P.S	Solution Architect	Professional Services

Name(s) of the PP Faculty: Dr.Mohammed Mujeer Ulla

Key Words: Electricity generation analysis, BDB tool, Optimizing resources, Analytical tool, Insights Trends and patterns, Data from multiple sources, Real-time monitoring, Performance optimization, Cost reduction, Environmental impact, Maximizing resources,

Project Areas: Business Analytics

Abstract:

The electricity generation analysis using BDB tool is a technique that can help electricity generation plants optimize their resources. The BDB tool is a powerful analytical tool that provides insights into electricity generation trends, patterns, and data from multiple sources. It can be used to track electricity generation and consumption in real-time, allowing plants to monitor and optimize their performance. This analysis helps identify areas of inefficiency and helps to determine the best strategies for reducing costs and improving the overall efficiency of electricity generation. Additionally, the BDB tool provides insights into the environmental impact of electricity generation, allowing plants to make informed decisions about their energy generation and consumption. This abstract highlights the importance of using the BDB tool for electricity generation analysis, as it can help plants maximize their resources, improve efficiency, and reduce their environmental impact.

Darren

Signature(s) of Student(s)

Date: 25/05/2023

Signature of PP Faculty

Date: 26/05/2023

CIN: U72900KA2015PTC079034
Email: accounts@bdb.ai

20th April, 2023

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Darren Andrew Dcunha** has successfully completed his Internship as **Intern – Data Analyst** in Professional Services Department at BizViz Technologies Pvt. Ltd., Bangalore under the guidance of Mr. Gaurav Mathur from **16th January, 2023 till 17th April 2023.**

We wish him all the best.

Yours sincerely,
For BizViz Technologies Pvt. Ltd.

Sonal Pathak
Recruitment & HR Manager

CIN: U72900KA2015PTC079034
Email: accounts@bdb.ai

27th December,2022

BY AND BETWEEN:

BizViz Technologies Pvt. Ltd (BDB),a company incorporated under the provisions of the Companies Act, 2013 and having its registered office at #15, 2nd Floor, Susmishi Complex, 9th Main, 13th Cross, Sector 6, HSR Layout Bangalore- 560102, India (hereinafter referred to as the "Company", which expression shall, unless repugnant to the context or meaning thereof, be deemed to mean and include its successors in interest, administrators and permitted assigns), of the FIRST PART and **Darren Andrew Dcunha** residing at **Bangalore, Karnataka**

Dear **Darren,**

We are pleased to accept your request for internship & accept you as an **Intern - Professional Services** with **BizViz Technologies Pvt. Ltd (BDB)**. You are expected to accept this offer at the earliest. Your joining date is set at **16th January,2023 for the duration of 3 months w.e.f the date of joining;** the terms and conditions of your employment with the Company are set forth below. Your employment is contingent upon your medical fitness and agreement to these terms and conditions, as evidenced, by your signing the acknowledgment copy of this letter and returning it to us before you join the Company or when this offer letter is given to you.

Business of Company -BizViz Technologies Pvt. Ltd (BDB). is a Product Company and has developed a modern, next generation Analytics Platform to give 360°view of data. BDB competes globally with top BI & Analytics companies of the world therefore Company seeks to provide the highest level of consulting service to its customers by recruiting, locating, training and retaining employees who have and/or will develop the high level of computer programming, engineering and consulting skills necessary to fully service its customers and continue to develop the different products that it has. Being a unique India based Product Development Company; BDB invests substantial knowledge and resources to train its employees and make them outstanding professionals who can compete at the global level. Moreover, the Company, also at great expense, fosters the development and maintenance of personal contacts and relationships between its sales/marketing personnel and its customers. To protect these investments, and the Company's interests for which they are made, the terms and conditions of your employment expressly limits use and disclosure of confidential and proprietary information, soliciting our employees to leave their employment, and competing against us to market similar services to our customers during your employment and for Three years following termination of your employment. You agree that these provisions are reasonable and will not prevent you from obtaining employment after your employment with the Company ends.

Employee Initials Darren

Place of Employment -

Your place of posting will be **Bangalore.** The company reserves its right to transfer you to any place, in India or Abroad or its subsidiary or associate company. You shall abide by the rules and regulations pertaining to those of the company.

First 3 months -A product group or project would be allocated to you on completion of the first three months dependent on your potential and ability to grasp different technologies and availability of open positions. An internal test and discussions would be conducted on completion of these three months to ensure that you have not only picked up the development techniques but also to understand if you have learnt how to follow different procedures and internal processes of BDB. Once you are placed in a project, you would be required to come only 5 days a week.

Note: *In case you fail in any of the above-mentioned stages, you would be given additional three months to clear both the stages. In case you are still unable to perform as according to the expectations of the Company post completion of these three months, the management will discuss alternative options with you or might even ask you to look at options outside of BDB. In the last 8 years of our relationship with new trainees, our success ratio has always been at 99%.*

Compensation and Benefits

This will be an Unpaid Internship; compensation would be learning experience, which you take away at the end of three months.

Hours of work:

- A working day shall comprise of nine (9) hours inclusive of an hour's lunch break. If and whenever need arises, you are required to stretch beyond these 9 hours.
- As according to business requirements, you may be expected to work out of our client's office/site within India. During such a deployment, you will be required to align your daily working hours and/or regular work week as per the client's working norms.
- Company expects you to work 45 productive hours a week during your training period and 40 productive hours a week, after the training period.
- Lunch time, Tea-Coffee breaks, Social Media breaks etc. are excluded from the count of productive hours

Duration of Internship: Initially Internship duration will be of 3 months, w.e.f joining date.

Employee Initials Parveen

Counterparts. This agreement may be executed in multiple counterparts, each of which shall have the force and effect of an original.

General Instructions & Guidelines:

- All terms and conditions of your employment including compensation and benefits is a matter of strict confidence and you shall maintain it that way at all times.
- You shall not be involved in any act of maligning the reputation of the Company, its founders and employees. This includes restricting yourself from posting unwanted pictures, writing comments etc. on any social media or public portals. In the event that we get to know of any such act from your end, you will be solely responsible for the consequences thereafter including legal action. The same applies if you are indirectly involved or are aware of such acts.
- You will keep us informed of any change in your residential address, contact details and any other critical information.

You are requested to read through the entire document, attest your signature on all pages and send us back the signed copy of this letter as a token of your acceptance of this offer. You can send us an acceptance email as soon as you receive this offer letter.

Looking forward to a long and rewarding association with BizViz Technologies!!

Yours sincerely,

For BizViz Technologies Pvt. Ltd. (BDB)

BizViz Technologies Pvt Ltd BANGALORE

Sonal Pathak

HR & Recruitment Manager
Date: 27th December,22

Agreed and accepted by me

Darren

Darren Andrew Dcunha

Date: 15/01/2023

Employee Initials _____

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

(Private University Estd. in Karnataka State by Act No.41 of 2013)

PU/CSE/INT/0460/2022 **27th December 2022**

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Darren Andrew Dcunha** (20191COM0045) is S/o **Mr. Naveen Paul Dcunha** is studying in **7th** Semester of B. Tech (Computer Engineering) program in the School of Engineering of this University during the academic year 2022-23.

This certificate is issued on request made by the student to apply for Internship program at **"Biz Viz Technology".**

27/12/2022

Dr. Kalaiarasan C
Assoc.Dean School of -CSE & IS
Presidency University, Bengaluru-560064
Email: kalaiarasan@presidencyuniversity.in

Electricity Generation Analysis using BDB tool

for Conference Proceedings Published by IEEE[1]

Darren Andrew Dcunha
Student,
Department of Computer Engineering,
Presidency University, Bangalore, India

Abstract:
The electricity generation analysis using BDB tool is a technique that can help electricity generation plants optimize their resources. The BDB tool is a powerful analytical tool that provides insights into electricity generation trends, patterns, and data from multiple sources. It can be used to track electricity generation and consumption in real-time, allowing plants to monitor and optimize their performance. This analysis helps identify areas of inefficiency and helps to determine the best strategies for reducing costs and improving the overall efficiency of electricity generation. Additionally, the BDB tool provides insights into the environmental impact of electricity generation, allowing plants to make informed decisions about their energy generation and consumption. This abstract highlights the importance of using the BDB tool for electricity generation analysis, as it can help plants maximize their resources, improve efficiency, and reduce their environmental impact

Key Words: Electricity generation analysis, BDB tool, Optimizing resources, Analytical tool, Insights Trends and patterns, Data from multiple sources, Real-time monitoring, Performance optimization, Cost reduction, Efficiency improvement, Environmental impact, Maximizing resources, Energy generation and consumption, Resource optimization, Environmental impact reduction

Introduction

Electricity generation analysis is a critical aspect of the energy industry that involves evaluating the performance and efficiency of power plants. With increasing global demand for electricity, the need for efficient and reliable electricity generation has become more crucial than ever. As a result, there is a growing interest in using data analytics tools and techniques to optimize electricity generation and reduce costs.

Electricity generation analysis involves gathering data on power generation, consumption, and distribution, and using this data to evaluate the efficiency of power plants. The data collected may include factors such as the amount of fuel consumed, the amount of electricity generated, and the cost of production. The analysis can be done in real-time, providing power plant operators with immediate insights into their operations and allowing them to make informed decisions to optimize their processes.

Data analytics tools such as the BDB (Balance of Data) tool can be used to monitor and analyze the performance of power plants in real-time, providing insights into areas of inefficiency and identifying opportunities for optimization. By identifying areas of inefficiency, power plants can implement changes to their operations that can result in significant cost savings and improved performance. Additionally, electricity generation analysis can help power plants reduce their environmental impact by identifying areas where energy is being wasted or by optimizing the use of renewable energy sources.

In summary, electricity generation analysis is a critical aspect of the energy industry that can help power plants optimize their resources, improve efficiency, reduce costs, and reduce their environmental impact. By leveraging data analytics tools and techniques, power plants can make informed decisions and remain competitive in a rapidly evolving energy landscape.

Technology Used:

Big Data Analytics: Big data analytics tools are used to process and analyze the large amounts of data collected from power plants. These tools can identify patterns, trends, and areas of inefficiency, allowing power plant operators to optimize their operations.

Machine Learning: Machine learning algorithms are used to identify patterns and predict future outcomes based on historical data. In electricity generation analysis, machine learning can be used to predict energy consumption and production, identify potential issues before they occur, and optimize power plant operations.

Visualization Tools: Visualization tools are used to present data in a way that is easy to understand, allowing power plant operators to quickly identify areas of inefficiency and opportunities for optimization.

2.4 Industrial Scope

Electricity generation analysis has a significant industrial scope, as it is relevant to a range of industries and sectors that rely on electricity for their operations. Here are some examples of industries where electricity generation analysis is particularly relevant:

Power Generation: The power generation industry is one of the most significant areas where electricity generation analysis is used. Power plants use data analytics tools to optimize their operations, improve efficiency, and reduce costs.

Manufacturing: The manufacturing industry relies heavily on electricity for its operations, and electricity generation analysis can help optimize energy usage and reduce costs. By identifying areas where energy is being wasted, manufacturers can make changes to their operations that can result in significant savings.

Transportation: The transportation industry, particularly the electric vehicle segment, is experiencing significant growth, and electricity generation analysis can help optimize the use of electric vehicle charging stations, reducing costs and improving the efficiency of the charging process.

Healthcare: The healthcare industry relies heavily on electricity for its operations, particularly in hospital settings. Electricity generation analysis can help hospitals optimize energy usage, reducing costs and improving the sustainability of their operations.

Agriculture: The agriculture industry is also increasingly reliant on electricity for its operations, particularly in areas such as irrigation and crop processing. Electricity generation analysis can help farmers optimize energy usage, reducing costs and improving the efficiency of their operations.

In summary, electricity generation analysis has a broad industrial scope, with applications in a range of industries and sectors that rely on electricity for their operations. By leveraging data analytics tools and techniques, these industries can optimize their energy usage, reduce costs, and improve their environmental sustainability.

2.5 Goal

The goal of electricity generation analysis is to optimize the performance and efficiency of power plants by using data analytics tools and techniques. By analyzing data related to power generation, consumption, and distribution, electricity generation analysis can help identify areas of inefficiency and opportunities for optimization. The ultimate goal is to improve the overall performance of power plants, reduce costs, and increase their environmental sustainability.

Project Work Flow

3.1 Problem Statement

The need to optimize the performance, efficiency, and cost-effectiveness of power generation processes. Despite significant advancements in technology, power plants still face challenges in maintaining optimal performance and efficiency. Some of the challenges include:
Energy Waste: Power plants may waste energy due to factors such as equipment malfunctions, inefficient processes, or inadequate maintenance.
Environmental Impact: Power generation processes can have a significant environmental impact, particularly with regard to greenhouse gas emissions.
Cost: Power plants face significant cost pressures, with fuel costs, maintenance costs, and labor costs all contributing to overall operating expenses.
To address these challenges, power plants need to leverage data analytics tools and techniques to optimize their operations and reduce costs. The use of data analytics can help identify areas of inefficiency,

allowing power plants to make informed decisions to improve their processes and reduce energy waste. Additionally, data analytics can help power plants optimize the use of renewable energy sources, reducing their environmental impact. Overall, the problem statement for electricity generation analysis is the need to optimize power generation processes to improve efficiency, reduce costs, and minimize environmental impact.

4. Data Extraction and Preparation

4.1 Request API from EIA

Navigate to EIA (Environment Impact Analysis)

We extracted data from the EIA (Environment Impact Analysis) website, which provides hourly electricity generation data for various plants in the USA.

This website provides data about electricity generated by various plants in USA on hourly basis

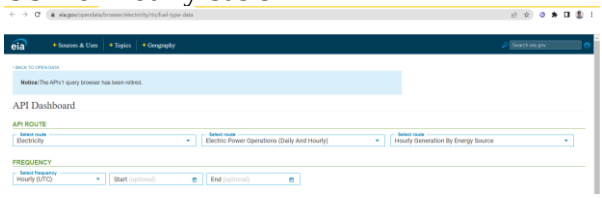


Fig.4.1 API Dashboard

4.2 Data Scraping

The data extraction was performed using a Python script that utilized the EIA API. The extracted data was cleaned, processed, and stored in a database for further analysis.

Code:

```
import requests
import datetime
import pandas as pd
from 4.2 Data import Label Encoder

def data extract():

    api_key =
'lS07rlQD1KkhGcZ9BMVfexEKONjRpb
YtUYJodOR8'
    #today = datetime.date.today() -
datetime.timedelta(days=4)
    url1 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start=2023-04-
11T00&end=2023-04-
11T12&sort[0][column]=period&sort[0][d
irection]=asc&offset=0&length=5000'
    url2 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start=2023-04-
11T13&end=2023-04-
11T23&sort[0][column]=period&sort[0][d
irection]=asc&offset=0&length=5000'
    url3 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start=2023-04-
12T00&end=2023-04-
12T12&sort[0][column]=period&sort[0][d
irection]=asc&offset=0&length=5000'
    url4 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start=2023-04-
12T13&end=2023-04-
12T23&sort[0][column]=period&sort[0][d
irection]=asc&offset=0&length=5000'
    url5 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start=2023-04-
13T00&end=2023-04-
13T12&sort[0][column]=period&sort[0][d
irection]=asc&offset=0&length=5000'
    url6 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start=2023-04-
13T13&end=2023-04-
13T23&sort[0][column]=period&sort[0][d
irection]=asc&offset=0&length=5000'
    url7 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
```

```
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start=2023-04-
14T00&end=2023-04-
14T12&sort[0][column]=period&sort[0][d
irection]=asc&offset=0&length=5000'
    url8 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start=2023-04-
14T13&end=2023-04-
14T23&sort[0][column]=period&sort[0][d
irection]=asc&offset=0&length=5000'

    le = LabelEncoder()
    rows = []
    for i in [1,2,3,4,5,6,7,8]:
        url = eval(f'url{i}')
        response = requests.get(url)

        if response.status_code == 200:
            data =
response.json()['response']['data']
            for d in data:
                date = d['period'] + '-01'  # add
day to format as YYYY-MM-DD
                Respondent_name =
d['respondent-name']
                Type_name = d['type-name']
                Value = d['value']
                Value_unit = d['value-units']
                rows.append({'YearMonth': date,
'respondent_name': Respondent_name,
'type_name': Type_name, 'value': Value,
'value_units': Value_unit})
        else:
            print(f'Request failed with status
code {response.status_code}.')

    if len(rows) == 0:
        print('No data available.')
        return pd.DataFrame()

    df = pd.DataFrame(rows)
    df = df.dropna()
    df['YearMonth'] =
pd.to_datetime(df['YearMonth'],
format='%Y-%m-%dT%H-%M')
    df['date']=df['YearMonth'].dt.strftime('%Y
-%m-%d')

    df['time']=df['YearMonth'].dt.strftime('%H
:%M')
    df = df.drop('YearMonth', axis=1)
    df['type_name_id'] =
le.fit_transform(df['type_name'])
    df['respondent_name_id'] =
le.fit_transform(df['respondent_name'])
    return df
```

modified code to extract code on daily basis:
code:

```
import requests
import datetime
import pandas as pd
import clickhouse_driver

def data_extract():

    api_key =
'lS07rlQD1KkhGcZ9BMVfexEKONjRpb
YtUYJodOR8'
    today = datetime.date.today() -
datetime.timedelta(days=2)

    url1 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start={today}T0
0&end={today}T12&sort[0][column]=peri
od&sort[0][direction]=asc&offset=0&leng
th=5000'
    url2 =
f'https://api.eia.gov/v2/electricity/rto/fuel-
type-
data/data/?api_key={api_key}&frequency
=hourly&data[0]=value&start={today}T1
3&end={today}T23&sort[0][column]=peri
od&sort[0][direction]=asc&offset=0&leng
th=5000'

    clickhouse_host =
'clickhouse.clickhouse'
    clickhouse_port = 9000
    clickhouse_user = 'darren_andrew'
```

```
clickhouse_database='db_darren_andrew'
    clickhouse_password = 'Tg4246'

    connection = clickhouse_driver.connect(
        host=clickhouse_host,
        port=clickhouse_port,
        user=clickhouse_user,
        database=clickhouse_database,
        password=clickhouse_password
    )

    cursor = connection.cursor()

    # Query ClickHouse tables to retrieve
labels
    cursor.execute('SELECT
respondent_name, respondent_name_id
FROM Electricity_respondent_id')
    respondent_labels = cursor.fetchall()

    cursor.execute('SELECT type_name,
type_name_id FROM Electricity_type_id')
    type_labels = cursor.fetchall()

    rows = []
    for i in [1, 2]:
        url = eval(f'url{i}')
        response = requests.get(url)

        if response.status_code == 200:
            data =
response.json()['response']['data']
            for d in data:
                date = d['period'] + '-01'  # add
day to format as YYYY-MM-DD
                respondent_name =
d['respondent-name']
                type_name = d['type-name']
                value = d['value']
                value_unit = d['value-units']
                rows.append({'YearMonth': date,
'respondent_name': respondent_name,
'type_name': type_name, 'value': value,
'value_units': value_unit})
        else:
            print(f'Request failed with status
code {response.status_code}.')

    if len(rows) == 0:
        print('No data available.')
        return pd.DataFrame()

    df = pd.DataFrame(rows)
    df = df.dropna()
    df['YearMonth'] =
pd.to_datetime(df['YearMonth'],
format='%Y-%m-%dT%H-%M')
    df['date'] =
df['YearMonth'].dt.strftime('%Y-%m-%d')
    df['time'] =
df['YearMonth'].dt.strftime('%H:%M')
    df = df.drop('YearMonth', axis=1)

    # Replace label encoding with
ClickHouse labels
    respondent_labels_dict =
dict(respondent_labels)
    df['respondent_name_id'] =
df['respondent_name'].map(respondent_lab
els_dict)
    type_labels_dict=dict(type_labels)

df['type_name_id']=df['type_name'].map(t
ype_labels_dict)
    return df
```

4.3 Deploy it into pipeline

the above code is written in python script component in Pipeline editor of BDB tool

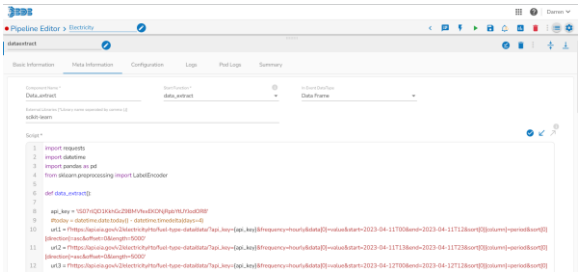


Fig.4.3.1 Python component(Data extract code)

To write the scrapped data into an Data Base , add a DB writer to the pipeline editor .

Before adding the DB writer we need add an Kafka event in between the python script component and DB writer

The output from the python script component is reflected in the kafka event below

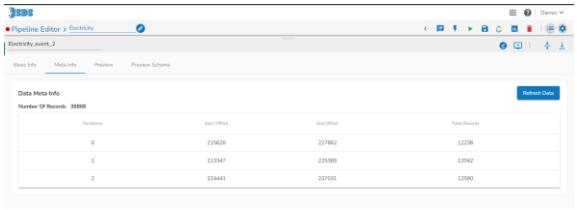


Fig.4.3.2 Kafka event (Data extract code)

From the above image we can infer that the python component has sent 36868 records from eia.
Here is a preview of the data depicted in the image below.

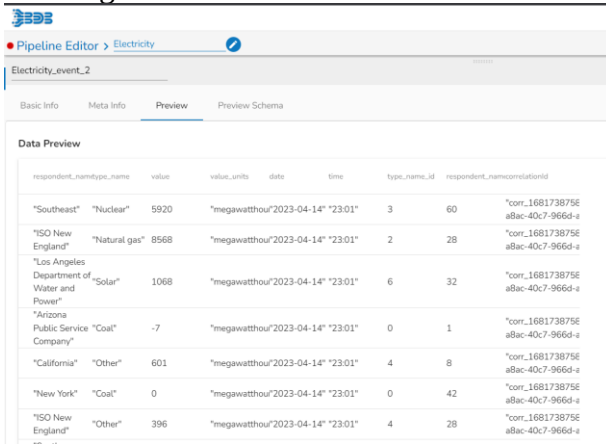


Fig 4.3.3 Preview of Extract data

The data is then written to a database with the help of the DB writer by providing the necessary credentials

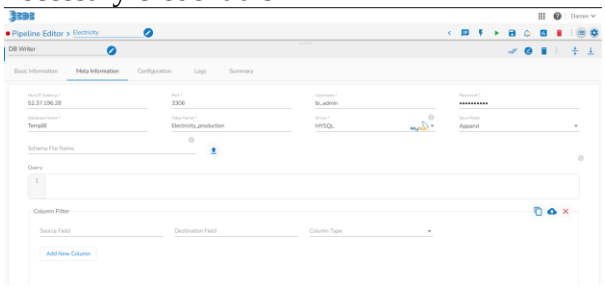


Fig 4.3.4 Configuration of Database

The below image shows that the data is stored in DataBase

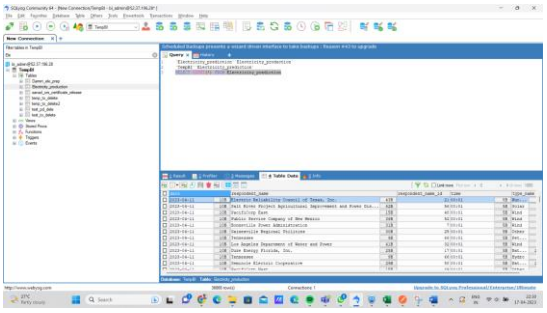


Fig 4.3.5 data is stored in DataBase

5. Training the model

Let's use this data to train the regression model:

This part of the workflow involves training a model using the above data.

5.1 Connect to the database

At first we need to connect the Database using the Data connector option from data centre module.
Here I'm selecting mysql as my dataconnector for this project.

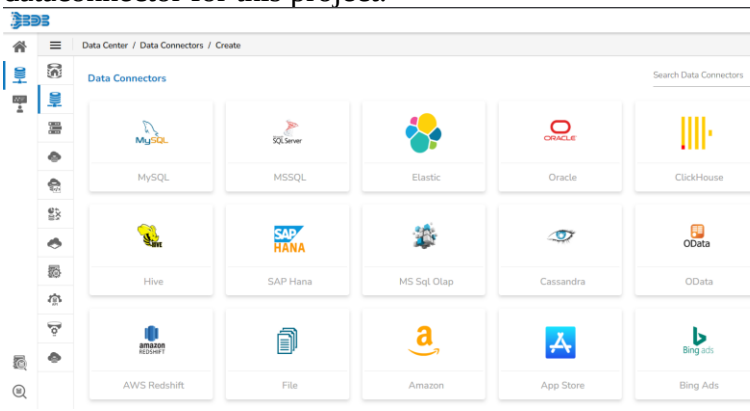


Fig.5.1.1 DataConnectors

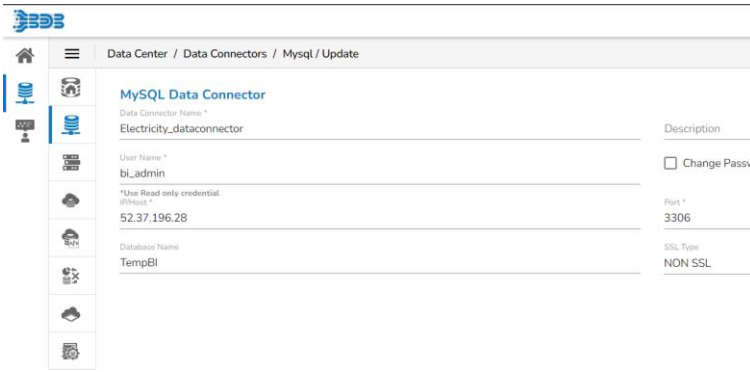


Fig.5.1.2 configuring DataConnector

Now let's create a new dataset from the newly created dataconnector.

The Below image is a view of dataset page. The following sql query is given to retrieve the data.

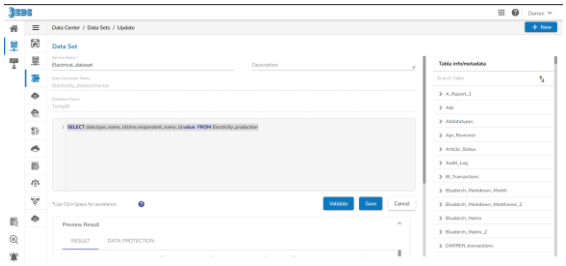


Fig.5.1.3 Configuring dataset

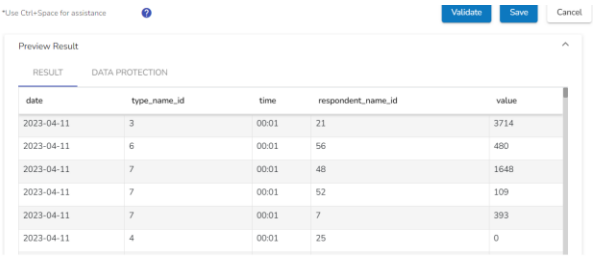


Fig.5.1.4 preview of data

Validate the above query and click on save.
From here we need to move to DS Labs.
At first we need to create a project and activate it.

5.2 Upload the Dataset toAutoML

At first create a project in DS labs and then Activate it.

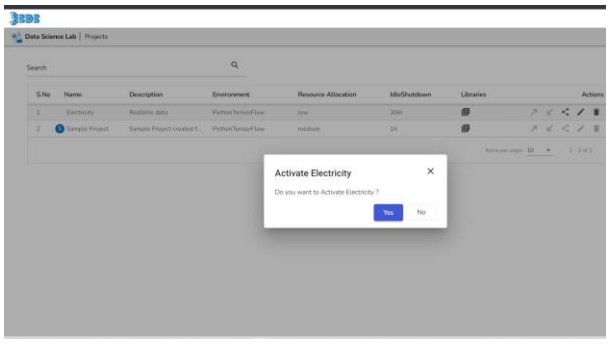


Fig.5.2.1 Configuring DS labs

Click on the dataset tab

Click on add dataset. Select the dataset that was created using the above steps.
Once the dataset is added to the DS Lab click on create experiment.

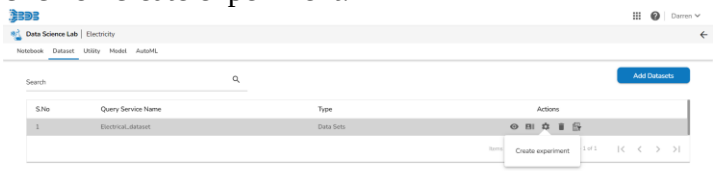


Fig.5.2.2 Creating AutoML Experiment

It will take you to a page to configure the automl experiment. Give a name and the target column (Here I'm selecting value as my target column).

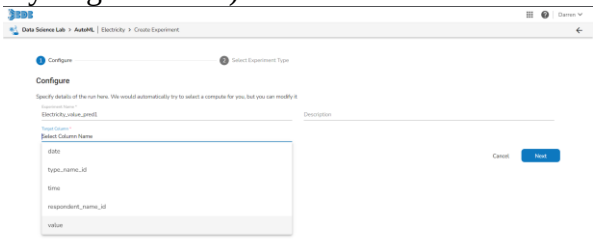


Fig.5.2.3 Configuring AutoML

Click on next. In this page we have an option to select classification or regression (
Here I will be selecting regression.)
From here the experiment will be created and after few minutes the automl will complete its execution (as seen from the below image status==completed).

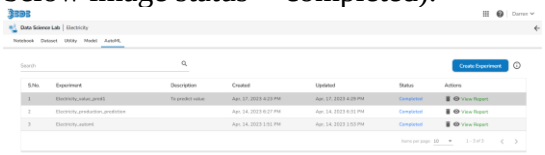


Fig.5.2.4 Created AutoML experiment
The automl feature in DS Labs recommends the best model for the given dataset. The below image gives a brief explaination it.

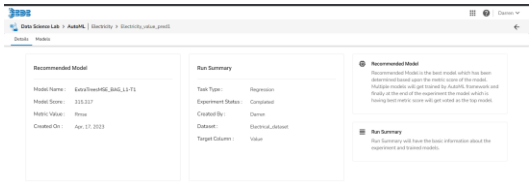


Fig.5.2.5 Recommended model

Moreover the automl suggests a total of 3 models which could be suitable for the dataset with its rmse score

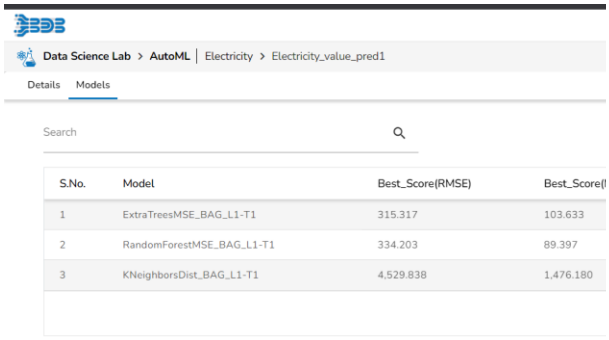


Fig.5.2.6 List of Models

We can also get information of this model starting with feature importance, Regression stats, Feature dependence and so on.

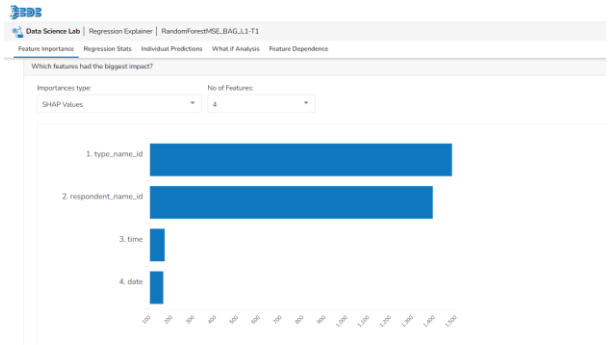


Fig.5.2.7 Model explainer

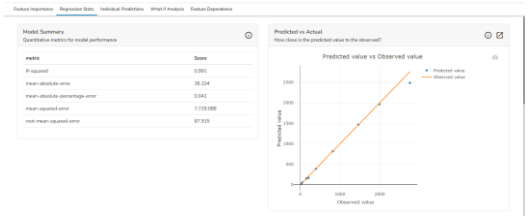


Fig.5.2.8 Model summary

From here the model needs to be uploaded to the pipeline.
Navigate to the Model tab change the filters to all from registered.
From here we could see the automl experiment.
Expand the models and click on the register button.

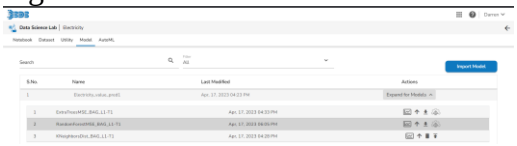


Fig.5.2.9 register model

5.3 Upload the model to pipeline

In the pipeline drag and drop the automl component
Configure the component by selecting the DS Labproject and the registered model

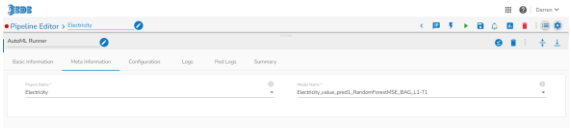


Fig 5.3.1 Configure AutoML component
Here is a preview of the data depicted in the image below

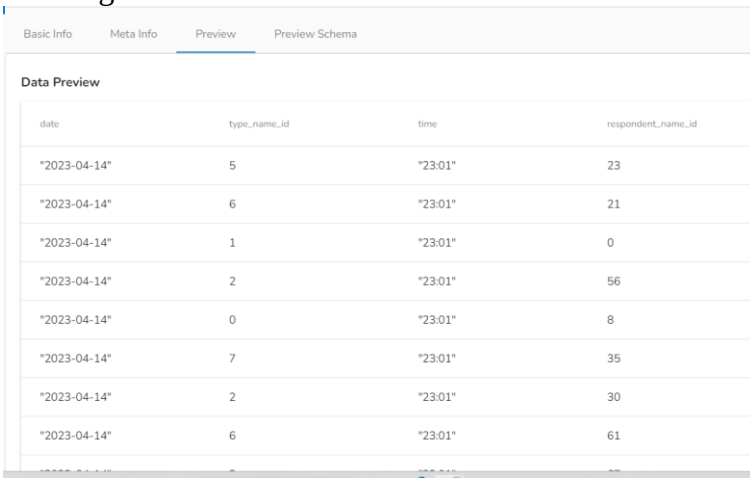


Fig.5.3.2 Output of the model

As we can see in the above image that the AutoML component has added a new column “predictions” where all the predicted values are stored

The above data is then stored in another DB writer

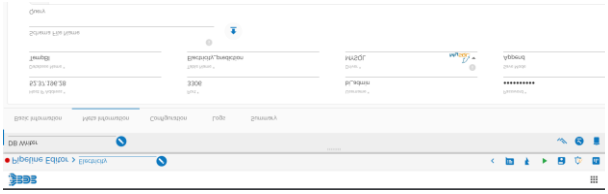


Fig.5.3.3 Configuring DB writer

6. Prepare dataset for next 6 days:
Once the model is trained, we can create a dataset to be input into the AutoML component, which will generate predicted values as the output.

Code:

```
import pandas as pd
from datetime import date, timedelta

def replicate_dataframe_with_dates(df):
    # Get yesterday's date
    yesterday = date.today() -
timedelta(days=1)

    # Copy the original DataFrame multiple
times with different dates starting from
yesterday
    new_dates =
pd.date_range(start=yesterday, periods=6,
freq='D')
    copied_dfs = []

    for new_date in new_dates:
        copied_df = df.copy()
        copied_df['date'] =
new_date.strftime('%Y-%m-%d')
        copied_dfs.append(copied_df)

    # Concatenate the modified DataFrames
with the original DataFrame
    combined_df = pd.concat([df] +
copied_dfs, ignore_index=True)

    # Drop unnecessary columns and
reorder columns
    combined_df =
combined_df.drop(['type_name',
'respondent_name', 'value_units', 'value'],
axis=1)
    combined_df = combined_df[['date',
'type_name_id', 'time',
'respondent_name_id']]

    return combined_df
```

The output generated from the previous code is subsequently fed into the AutoML component within the pipeline. The resulting output of the AutoML component contains the predicted value. Afterwards, the data is transmitted to another Python script. The following script prepares the data for visualization purposes before storing it in a database.

Code:

```
import requests
import datetime
import pandas as pd
import clickhouse_driver

def data_combine(df):


    clickhouse_host =
'clickhouse.clickhouse'
    clickhouse_port = 9000
    clickhouse_user = 'darren_andrew'

clickhouse_database='db_darren_andrew'
    clickhouse_password = 'Tg4246'

    connection = clickhouse_driver.connect(
        host=clickhouse_host,
        port=clickhouse_port,
        user=clickhouse_user,
        database=clickhouse_database,
        password=clickhouse_password
    )

    cursor = connection.cursor()

    # Query ClickHouse tables to retrieve
labels
    cursor.execute('SELECT
respondent_name_id, respondent_name
FROM Electricity_respondent_id')
    respondent_labels = cursor.fetchall()
```

```
    cursor.execute('SELECT type_name_id,
type_name FROM Electricity_type_id')
    type_labels = cursor.fetchall()

    # Replace label encoding with
ClickHouse labels
    respondent_labels_dict =
dict(respondent_labels)
    df['respondent_name'] =
df['respondent_name_id'].map(respondent
_labels_dict)
    type_labels_dict=dict(type_labels)

df['type_name']=df['type_name_id'].map(t
ype_labels_dict)
    return df
```

The above data is stored in a Database. The below picture depicts the complete pipeline.

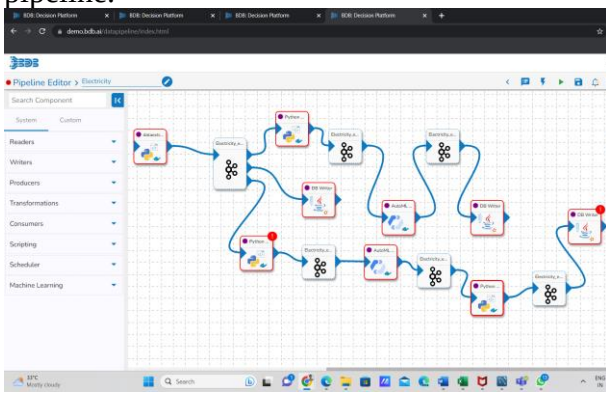


Fig.6 Pipeline editor

7. Forecasting:

Electricity generation plays a crucial role in ensuring the stability and efficiency of power systems. Accurate forecasting of electricity generation can aid in efficient load management, resource planning, and decision-making processes. In this project, we aim to develop an ESM forecast model to predict electricity generation for the next 24 hours.

For this project, we retrieved data from a ClickHouse database. The relevant columns included the date, time, and electricity generation values. To ensure consistency and handle missing values, we resampled the data to an hourly frequency. Additionally, we performed linear interpolation and backward filling to handle missing values and outliers. Exponential Smoothing (ESM) was chosen as the forecasting technique for this project. ESM is a time series forecasting method that captures trend, seasonality, and smoothing levels. To determine the best model parameters, we conducted cross-validation using TimeSeriesSplit. The evaluation metric used was the root mean squared error (RMSE), which provides a measure of the model's accuracy.

Code:

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.tsa.holtwinters import
ExponentialSmoothing
from sklearn.metrics import
mean_squared_error
from sklearn.model_selection import
TimeSeriesSplit
from clickhouse_driver import Client

def predict_time_series():
    client =
Client(host='clickhouse.clickhouse',
port='9000', database='db_darren_andrew',
user='darren_andrew', password='Tg4246')

    # Execute query
    query = "SELECT * FROM
Electricity_production"
    results = client.execute(query)
    df = pd.DataFrame(results,
columns=['date', 'respondent_name',
'respondent_name_id','time','type_name','ty
pe_name_id','value','value_units'])
    # Close database connection
    client.disconnect()
    df['timestamp'] =
pd.to_datetime(df['date'] + ' ' + df['time'])
    df = df.set_index('timestamp')

    # resample data to hourly frequency
    df = df.resample('H').sum()

    # handle missing values and outliers
```

```
    df = df.replace(0, np.nan)
    df = df.interpolate(method='linear',
limit_direction='both')
    df = df.fillna(method='bfill')
    df['timestamp'] = df.index
    df['date'] =
df['timestamp'].dt.strftime('%Y-%m-%d')
    df['time'] =
df['timestamp'].dt.strftime('%H:%M')
    df['timestamp'] =
df['timestamp'].dt.strftime('%Y-%m-%d
%H:%M')



    # split data into training and testing sets
    split_index = len(df) - 24
    train_data = df.iloc[:split_index]
    test_data = df.iloc[split_index:]

    # define parameter grid for exponential
smoothing
    alpha = np.arange(0.1, 1.0, 0.1)
    beta = np.arange(0.1, 1.0, 0.1)
    gamma = np.arange(0.1, 1.0, 0.1)

    # perform cross-validation to find best
model
    best_score, best_params = float("inf"),
None
    cv = TimeSeriesSplit(n_splits=5)
    for train_index, val_index in
cv.split(train_data):
        train = train_data.iloc[train_index]
        val = train_data.iloc[val_index]
        for a in alpha:
            for b in beta:
                for g in gamma:
                    try:
                        model =
ExponentialSmoothing(train['value'],
trend='add', seasonal='add',
seasonal_periods=24).fit(smoothing_level
=a, smoothing_slope=b,
smoothing_seasonal=g)
                        y_pred =
model.forecast(steps=len(val))
                        score =
np.sqrt(mean_squared_error(val['value'],
y_pred))
                        if score < best_score:
                            best_score, best_params
= score, (a, b, g)
                        mape =
np.mean(np.abs((val['value'] - y_pred) /
val['value'])) * 100
                        # calculate the accuracy
as a percentage
                        accuracy = 100 - mape
                    except:
                        continue

    # fit the best model on the entire dataset
    forecast =
ExponentialSmoothing(df['value'],
trend='add', seasonal='add',
seasonal_periods=24)
    model =
forecast.fit(smoothing_level=best_params[
0], smoothing_slope=best_params[1],
smoothing_seasonal=best_params[2])

    # make predictions for the next 24 hours
after the available data
    next_24_hours =
pd.date_range(start=df.index[-1] +
pd.DateOffset(hours=1), periods=24,
freq='H')
    next_24_hours =
pd.DatetimeIndex(next_24_hours)
    next_24_hours_df =
pd.DataFrame(index=next_24_hours)

next_24_hours_df['date']=next_24_hours_
df.index.strftime('%Y-%m-%d')

next_24_hours_df['time']=next_24_hours_
df.index.strftime('%H:%M')

next_24_hours_df['timestamp']=next_24_h
ours_df.index.strftime('%Y-%m-%d
%H:%M')
    next_24_hours_predictions =
model.forecast(steps=24)
    pred_df = pd.concat([df,
next_24_hours_df])

next_24_hours_df['test_predictions']=next
_24_hours_predictions
```

return next_24_hours_df

The output gives forecast for next 24 hrs

The dataset was split into training and testing sets, with the last 24 hours reserved for testing. Through cross-validation, we identified the best model parameters that yielded the lowest RMSE. The chosen model demonstrated excellent performance, capturing the underlying patterns in the electricity generation data.

```
RMSE 43753.71547566049
MAPE 2.8699996952065563
Accuracy 97.13000030479344
```

	date	time	timestamp	test_predicti
2023-05-15 00:00:00	2023-05-15	00:00	2023-05-15 00:00	1.304279e
2023-05-15 01:00:00	2023-05-15	01:00	2023-05-15 01:00	1.268141e
2023-05-15 02:00:00	2023-05-15	02:00	2023-05-15 02:00	1.212346e
2023-05-15 03:00:00	2023-05-15	03:00	2023-05-15 03:00	1.119792e
2023-05-15 04:00:00	2023-05-15	04:00	2023-05-15 04:00	1.008560e
...	...	...	...	
2023-05-15 19:00:00	2023-05-15	19:00	2023-05-15 19:00	9.807618e
2023-05-15 20:00:00	2023-05-15	20:00	2023-05-15 20:00	1.010204e
2023-05-15 21:00:00	2023-05-15	21:00	2023-05-15 21:00	1.045572e
2023-05-15 22:00:00	2023-05-15	22:00	2023-05-15 22:00	1.044087e
2023-05-15 23:00:00	2023-05-15	23:00	2023-05-15 23:00	1.046514e

[24 rows x 4 columns]

Fig.7 ESM model performance

The best model parameters were used to fit the model on the entire dataset. This fitted model was then employed to make predictions for the next 24 hours after the available data. The forecasted values were obtained for each hour, allowing for a comprehensive understanding of electricity generation trends.

The ESM forecast model presented in this project provides valuable insights into electricity generation patterns. The accurate predictions for the next 24 hours allow for proactive decision-making, resource allocation, and load management in power systems. However, it is essential to note that the forecast model's performance may be influenced by factors such as data quality and the availability of historical data.

Overall, the developed ESM forecast model for electricity generation showcases the potential of time series analysis in optimizing power system operations and planning. It provides a foundation for future enhancements and applications in the field of electricity forecasting.

The provided code is integrated into a pipeline, where the resulting DataFrame (next_24_hours_df) is stored in a ClickHouse database.

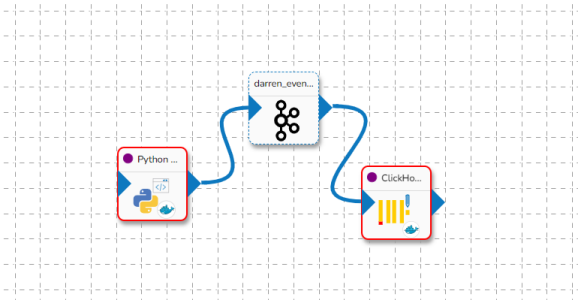


Fig 7.1 Forecasting pipeline

The provided code is integrated into a pipeline, where the resulting DataFrame (**yesterday_data**) is stored in a ClickHouse database.

To evaluate the performance and compare forecasted values with the actual values, the code snippet provided prepares the original data. Later, a SQL join query can be utilized to combine both tables for further analysis.

```
import datetime
import pandas as pd
import numpy as np
from clickhouse_driver import Client
def predict_time_series():
    client =
Client(host='clickhouse.clickhouse',
port='9000', database='db_darren_andrew',
user='darren_andrew', password='Tg4246')

    # Execute query
    query = 'SELECT * FROM
Electricity_production'
    results = client.execute(query)
    df = pd.DataFrame(results,
columns=['date', 'respondent_name',
'respondent_name_id','time','type_name','ty
pe_name_id','value','value_units'])
    # Close database connection
    client.disconnect()
    df['timestamp'] =
pd.to_datetime(df['date'] + ' ' + df['time'])
    df = df.set_index('timestamp')

    # resample data to hourly frequency
```

```
    df = df.resample('H').sum()

    # handle missing values and outliers
    df = df.replace(0, np.nan)
    df = df.interpolate(method='linear',
limit_direction='both')
    df = df.fillna(method='bfill')
    df['timestamp'] = df.index

df['date']=df['timestamp'].dt.strftime('%Y-
%m-%d')

df['time']=df['timestamp'].dt.strftime('%H:
%M')

df['timestamp']=df['timestamp'].dt.strftime
('%Y-%m-%d %H:%M')
    # Get yesterday's date
    yesterday = datetime.datetime.now() -
datetime.timedelta(days=2)
    yesterday_start =
yesterday.replace(hour=0, minute=0,
second=0)
    yesterday_end =
yesterday.replace(hour=23, minute=59,
second=59)

    # Filter dataframe for yesterday's data
    yesterday_data = df.loc[(df.index >=
yesterday_start) & (df.index <=
yesterday_end)]

    return yesterday_data
```

The provided code is integrated into a pipeline, where the resulting DataFrame (yesterday_data) is stored in a ClickHouse database.

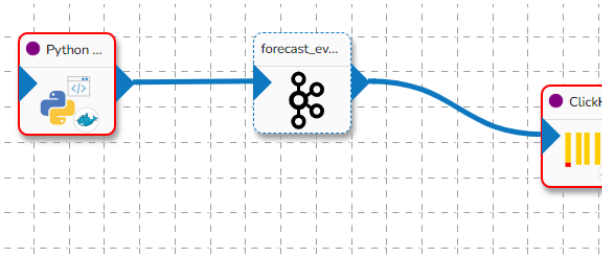


Fig.7.2 pipeline editor

8 Visualization

Let's visualize by plotting charts and get more insights of the data. From here we'll be using business story module of the BDB tool.

8.1 First collect the data using datasets from datacenter module.

Using the existing data connector click on the '+' icon and then create a new data store.

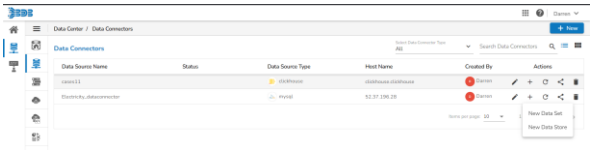


Fig 8.1 Creating a datastore

Then write an SQL query

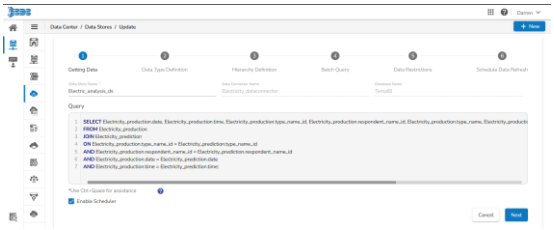


Fig 8.2 SQL query for datastore

Categorize the fields based on measures, dimensions and time.
Here I dragged date and time field from Dimensions to time.

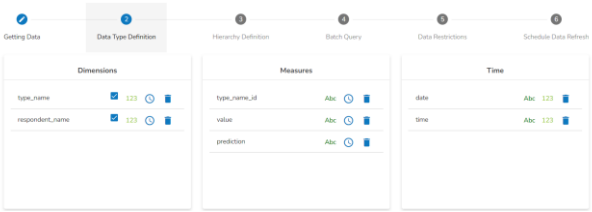


Fig.8.3 Datatype defination

On clicking next button, we can add hierarchy to the data,
By just date to the hierarchy, it automatically split date into year month and day. Moreover we can add another drill and add the respective fields respondent name and type name .

Since we do not require batch query and data retriction we can skip these steps.
At last we have Schedule data refresh

Since I want to refresh the data everyday since data is added to the data base everyday.
Set scheduler to daily and check tick on refresh now.

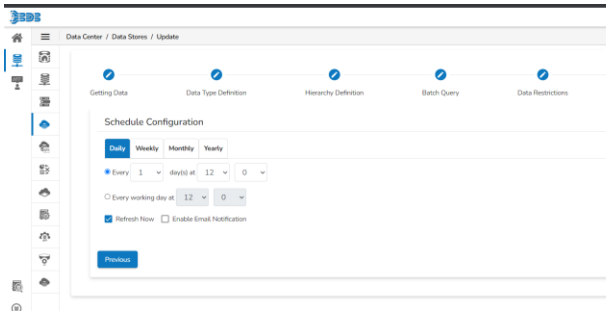


Fig 8.3 Schedule

Click on finish and now the data store will be created.

8.2 Business Story

Let's use this data store to create a business story.
In the home screen navigate to the business story.
A create story page will appear give a name and select the data store that we created using the above steps.

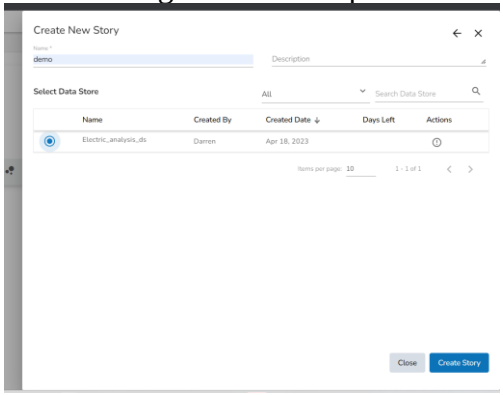


Fig 8.2.1 Creating a story

Click on create story.

The below image is the interface of the business story.

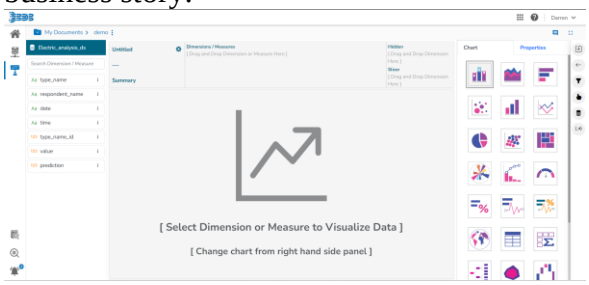


Fig 8.2.2 Story canvas

On the left plane we can see the fields of the data and on the right we can see the list of charts.

At first I'll plot a KPI.
On the right select the KPI chart then select the respondent_name field from the left pane.
In the below image we can see that a KPI has been plotted.

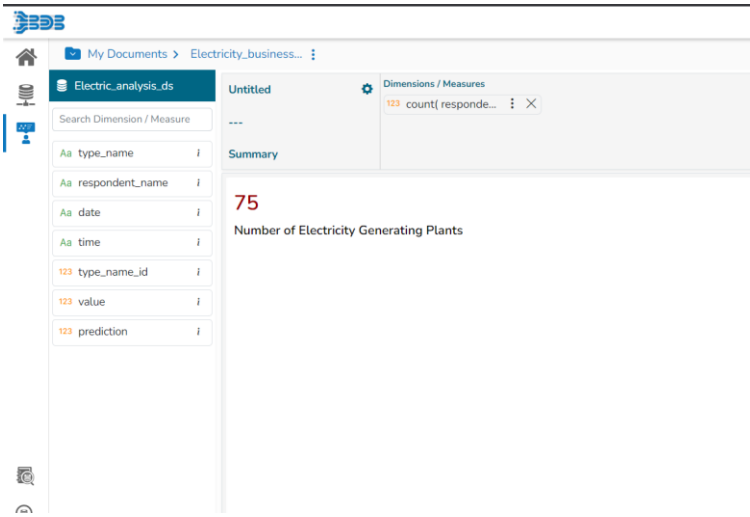
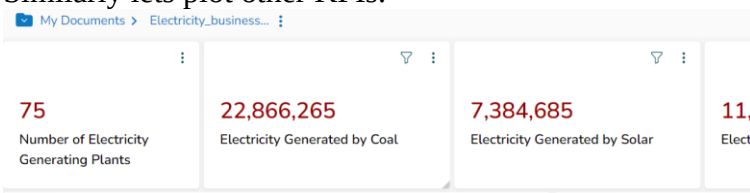


Fig.8.2.3 Ploting KPI's
Click on save.
Similarly lets plot other KPIs.



As we can see in total I have plotted 5 KPIs

Now let's plot a bar graph.
In this bar graph let us find the average electricity produced on each day.

Select date and value fields to the top pane.
Just beside the value there is a 3 dot icon, click on it and an window appears on the right side. Select the aggregation as mean.

Fig.8.2.3 configuring graphs

Then select the bar graph from the charts menu on the right pane.

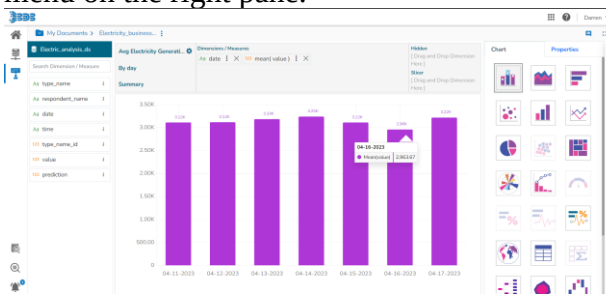


Fig.8.2.3 Bar Graphs

Similarly we can plot actual vs the predicted values from AutoML .
Select fields type_name, value and predicted as sum aggregation.

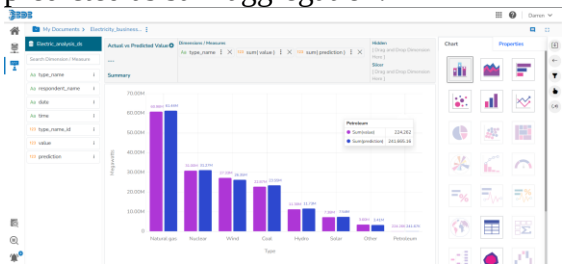


Fig.8.2.4 Actual VS Prediction

Now lets plot a horizontal bar graph.
Select fields respondent_name and value.
In the properties tab from the right pane.
Configure the order, orderby and set limit to 10.
The following graph will show the top 10 Generation plants .

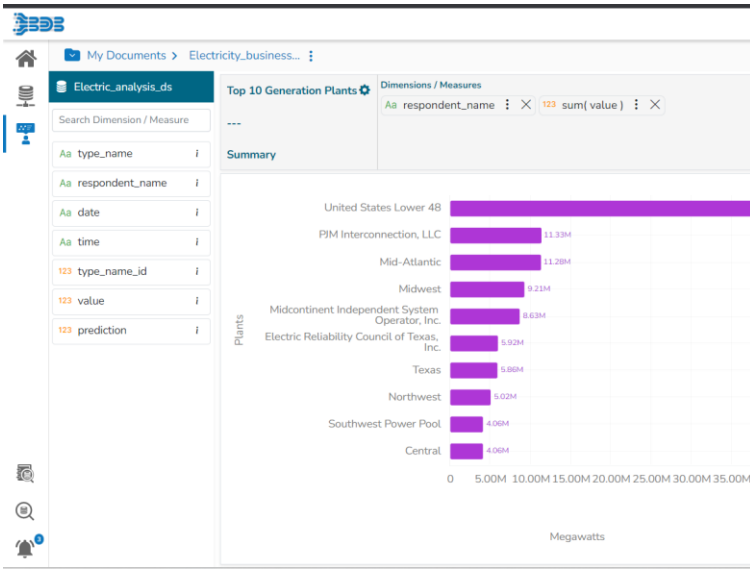


Fig.8.2.5 Respondent graph

Tree Map chart
Select fields type name and value.

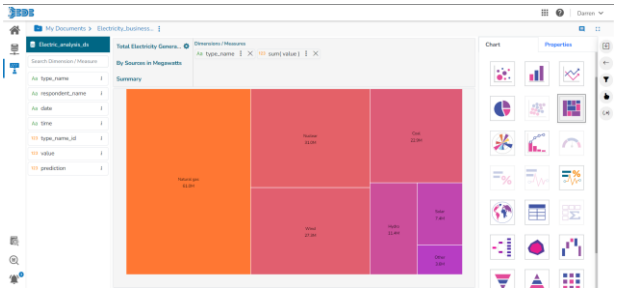


Fig.8.2.6 Tree map

Decomposition tree:
Select fields respondent_name , type_name and value.

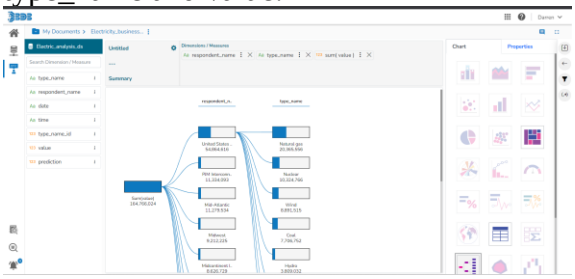


Fig.8.2.7 Decomposition

8.3 Business story for Prediction Data

Similarly lets create another business story to analyse Predictions done using Regression.
First lets create a Datastore using the below sql query.

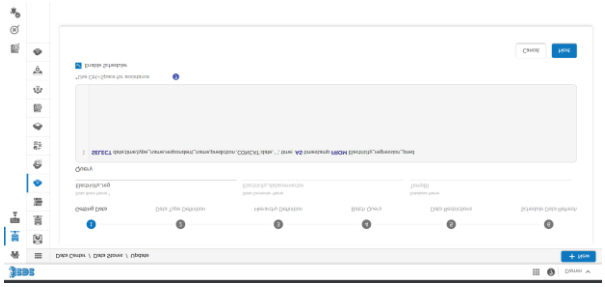


Fig.8.3.1 Datastore for regression data

Now create a story and select the above datastore.

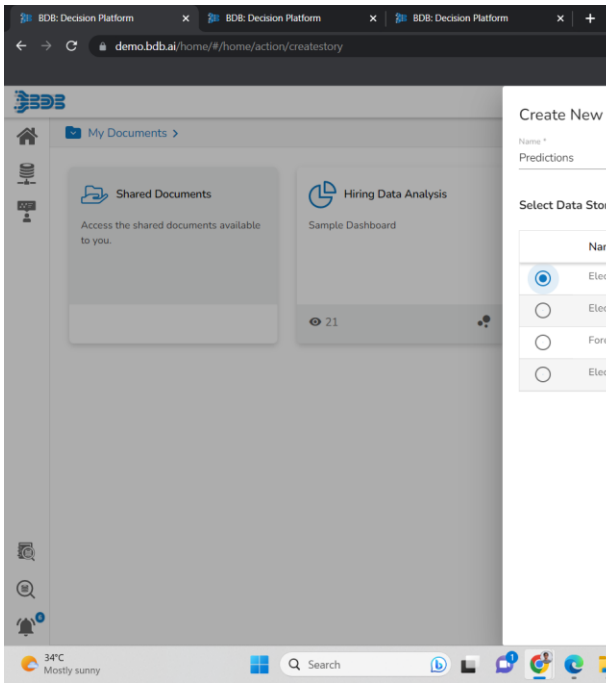


Fig.8.3.2 Creating story

Just like previous story let's plot few KPIs a column chart(Source vs Power generated), bar chart (Respondent vs Power generated) and a line chart(timestamp vs power generated)

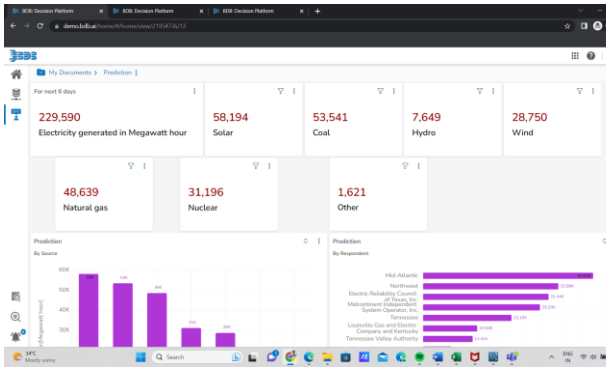


Fig.8.3.3 Story for Predicted values

The story displays information about the power that will be generated over the next 6 days. The model used to make these predictions is a random forest. A random forest is an ensemble learning method that combines multiple decision trees to make predictions. Decision trees are a type of machine learning algorithm that can be used to classify or predict values.

In the context of power generation, a random forest model can be used to predict the amount of power that will be generated over a given period of time. The model does this by training on historical data about power generation. The historical data includes information such as the time of day, the weather, and the amount of demand for power. The model uses this information to create a set of decision trees. Each decision tree makes a prediction about the amount of power that will be generated.

The random forest model then combines the predictions from the individual decision trees to make a final prediction about the amount of power that will be generated. The random forest model is a powerful tool that can be used to predict power generation. The model is accurate and reliable, and it can be used to make predictions over a long period of time.

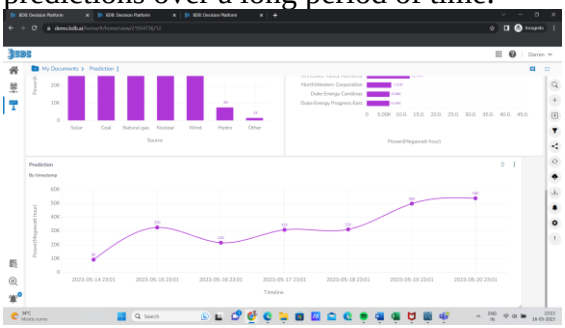


Fig.8.3.4 Timeline of Predicted values

8.4 Visualization for forecasting data (ESM)

The below two images creates a datastore that involves timeseries forecasting.

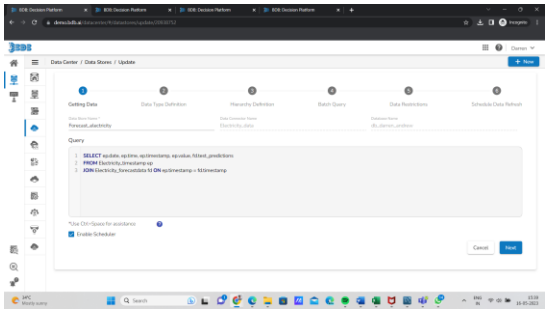


Fig. 8.4.1 Configuring Datastore for Forecasting data (Actual VS Forecast value)

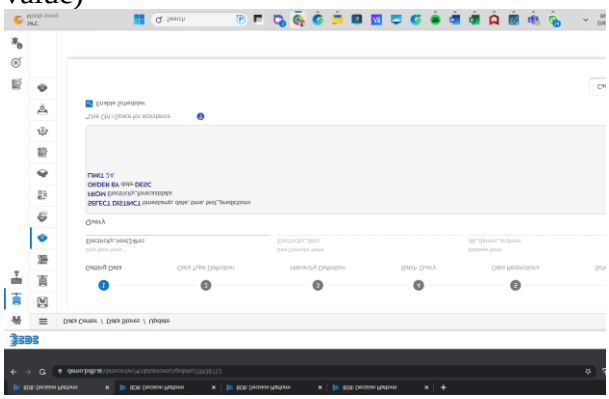


Fig. 8.4.2 Configuring Datastore for Forecasting data

To plot charts in the Dashboard Designer, the following steps are followed:

Navigate to the Dashboard Designer interface.
Click on the "New Dashboard" option to create a new dashboard.
On the right pane, select "Data Connectors" from the available options.
Click on the '+' icon near the data store to add a new data store.

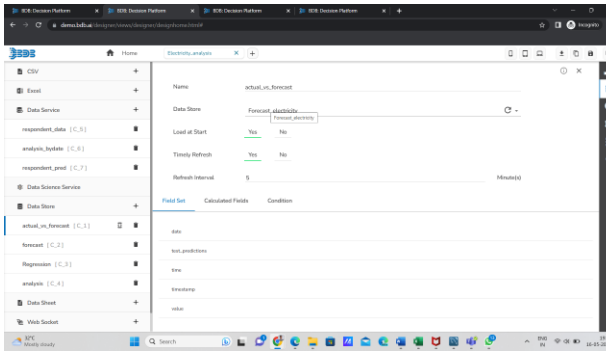


Fig.8.4.3 Configuring connections
Choose the appropriate data store from the options provided.
In the canvas area, drag and drop a timeseries chart component.

Select the desired dataset from the available options.
Choose the relevant fields from the dataset to be displayed on the chart.
Click on the "Preview" button to visualize and preview the chart.

From the above steps we could plot a forecasting graph.
The below image depicts the dashboard.

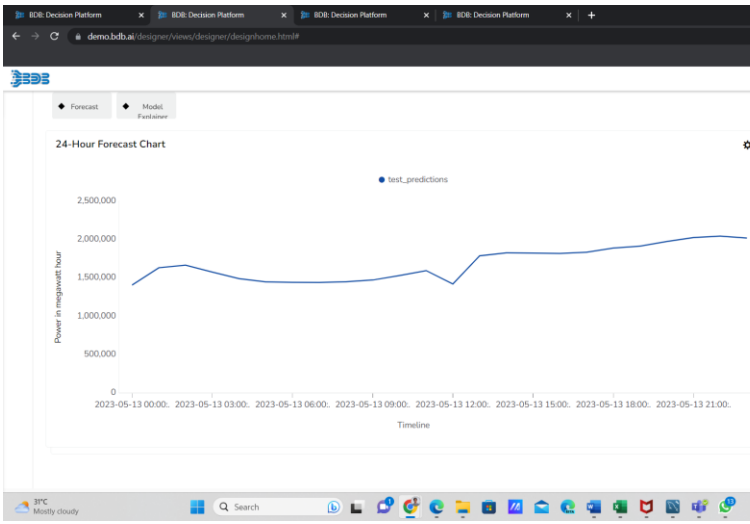


Fig.8.4.3 Forecast graph

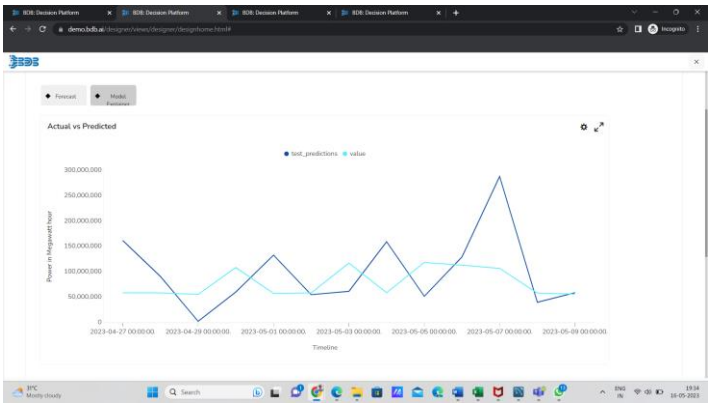


Fig.8.4.4 Actual VS Forecasted Graph

Similarly, in this dashboard, we can also plot the overview of the data and perform regression analysis. The steps mentioned earlier for plotting charts in the Dashboard Designer can be applied for these purposes as well. By selecting the appropriate chart types and datasets, we can create visualizations for data overview and regression analysis in the dashboard. The flexibility of the Dashboard Designer

allows us to customize and arrange the charts according to our specific requirements.

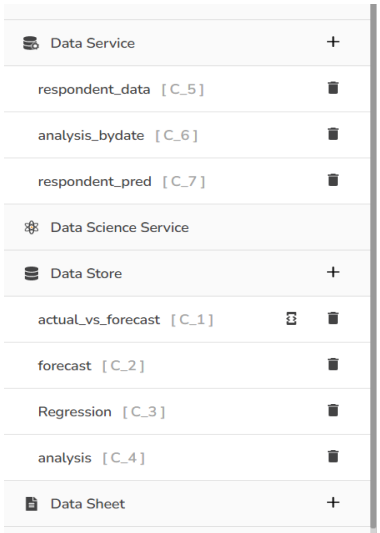


Fig.8.4.5 List of DataConnectors

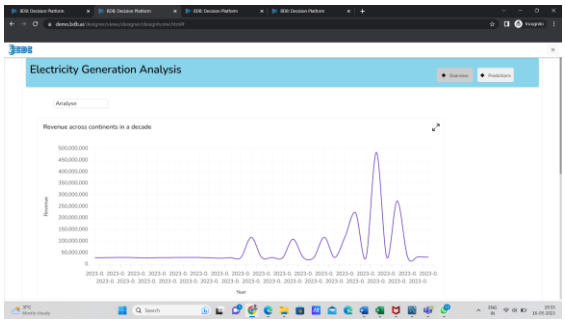


Fig.8.4.6 Overview of the data

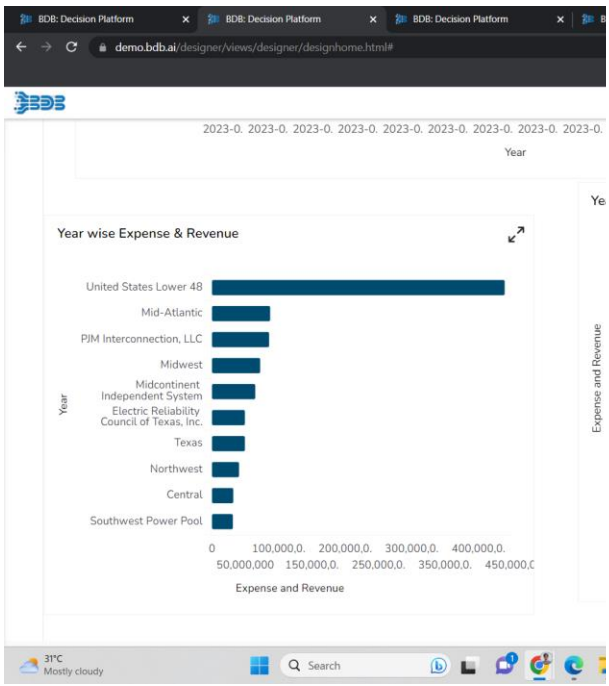


Fig.8.4.7 Overview of the data

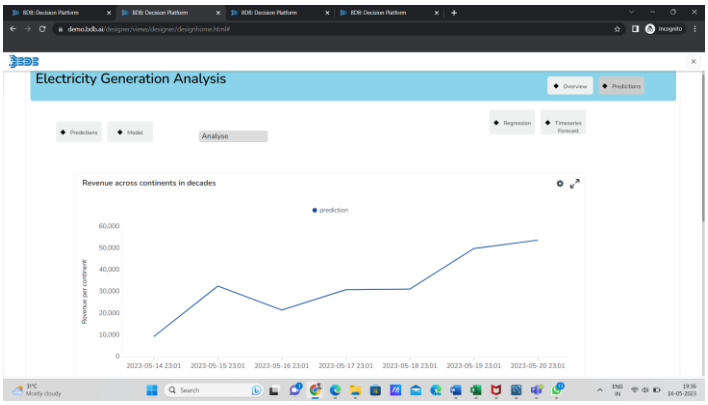


Fig.8.4.8 Regression analysis

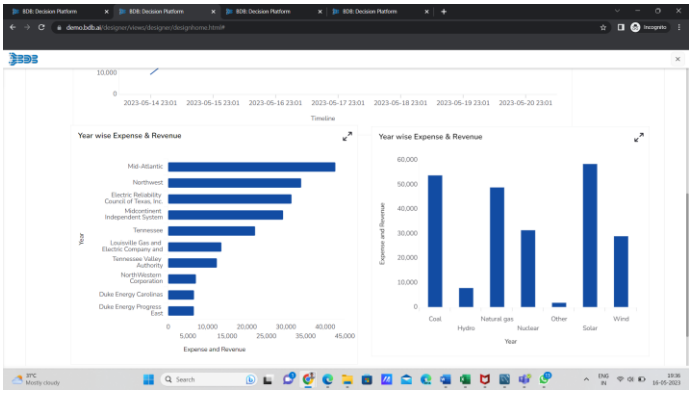


Fig.8.4.9 Regression analysis

Conclusion

In conclusion, the project aimed to develop a forecasting model for electricity generation using data extraction, data preparation, model training, and forecasting techniques. The project followed a pipeline-based approach to streamline the data processing and analysis tasks. Here are the key takeaways:

Data Extraction and Preparation: The project started by extracting electricity generation data from the EIA website using their API. A Python script was utilized to scrape the data, clean it, and store it in a database for further analysis. The data was then prepared by encoding categorical variables and structuring it in a suitable format.

Model Training: The prepared dataset was used to train a regression model using

AutoML in DS Labs. The model selection was automated, and the best-performing models were recommended based on their RMSE scores. The selected model was registered and integrated into the pipeline for further use.

Dataset Preparation for Forecasting: To generate predictions for the next six days, a dataset was created by replicating the existing data for multiple dates. The dataset was then fed into the AutoML component, which generated predicted values using the trained model. The predicted values were stored in a database for further analysis and visualization.

Forecasting: The final stage of the project focused on forecasting electricity generation for the next 24 hours using Exponential Smoothing (ESM) as the forecasting technique. The historical data from the database was resampled to an hourly frequency, and missing values were handled through interpolation and backward filling. The ESM model was trained using the resampled data, and its performance was evaluated using RMSE as the evaluation metric.

The project demonstrated the end-to-end process of extracting, preparing, training, and forecasting electricity generation data. By utilizing pipeline-based tools and techniques, it streamlined the workflow and automated certain steps, such as model selection and data replication. The project's outcomes can be valuable for load management, resource planning, and decision-making processes in the power industry.

Overall, the project showcased the potential of data-driven approaches in the field of electricity generation forecasting and provided insights into the accuracy and reliability of the developed models. Further improvements and refinements can be made to enhance the forecasting accuracy and explore additional forecasting techniques for better predictions in the future.

Glossary:

Forecasting: The process of making predictions or estimates about future values based on historical data and patterns.
Data Extraction: The process of retrieving data from various sources such as websites, databases, APIs, etc.
Data Preparation: The process of cleaning, transforming, and structuring raw data into a suitable format for analysis and modeling.
Model Training: The process of building a predictive model by feeding historical data to an algorithm to learn patterns and relationships.
Regression Model: A type of statistical model used to predict a continuous numerical value, such as electricity generation, based on input variables.
AutoML: Automated Machine Learning, a set of tools and techniques that automate the process of selecting, training, and evaluating machine learning models.
RMSE (Root Mean Square Error): A commonly used metric to measure the accuracy of a predictive model by calculating the square root of the average squared differences between predicted and actual values.
Dataset: A collection of structured data used for analysis and modeling.
Interpolation: A technique used to estimate missing values in a dataset by inferring values based on known data points.
Exponential Smoothing: A time series forecasting method that assigns exponentially decreasing weights to past observations to predict future values.
Workflow: The sequence of tasks or steps involved in completing a specific project or process.
Evaluation Metric: A measure used to assess the performance or accuracy of a predictive model, such as RMSE, MAE (Mean Absolute Error), or R-squared.

Fossil Fuels: Non-renewable energy sources, such as coal, oil, and natural gas, formed from the remains of ancient plants and animals.
Renewable Energy Sources: Energy sources derived from natural resources that are replenished over a short period, such as solar power, wind power, hydroelectricity, etc.

References:

https://www.bdb.ai/
https://towardsdatascience.com/
eia.gov/opendata/browser/electricity/rto/fuel-type-data
www.geeksforgeeks.org

Electricity generation analysis

ORIGINALITY REPORT

8%	7%	3%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Sydney Student Paper	1%
2	Pedro J. S. Cardoso, Jânio Monteiro, Nelson Pinto, Dario Cruz, João M. F. Rodrigues. "chapter 19 Application of Machine Learning Algorithms to the IoE", IGI Global, 2020 Publication	1%
3	github.com Internet Source	1%
4	cashflowinventory.com Internet Source	1%
5	www.enhgo.com Internet Source	<1%
6	Submitted to Monash University Student Paper	<1%
7	buildmedia.readthedocs.org Internet Source	<1%
8	www.section.io Internet Source	<1%

9	Submitted to Lappeenrannan teknillinen yliopisto Student Paper	<1%
10	Submitted to University of Bolton Student Paper	<1%
11	community.powerbi.com Internet Source	<1%
12	Submitted to University of Auckland Student Paper	<1%
13	www.questarter.com Internet Source	<1%
14	Fahad Bin Abdullah, Rizwan Iqbal, Falak Shad Memon, Sadique Ahmad, Mohammed A. El-Affendi. "Advancing Sustainability in the Power Distribution Industry: An Integrated Framework Analysis", Sustainability, 2023 Publication	<1%
15	ianwhitestone.work Internet Source	<1%
16	medium.com Internet Source	<1%
17	rdrr.io Internet Source	<1%
18	Hieu Huynh, Thomas J. Kelly, Linh Vu, Tung Hoang, Phuc An Nguyen, Tu C. Le, Emily A.	<1%

	Jarvis, Hung Phan. "Quantum Chemistry–Machine Learning Approach for Predicting Properties of Lewis Acid–Lewis Base Adducts", ACS Omega, 2023 Publication	
19	Submitted to University of Greenwich Student Paper	<1 %
20	lifesciencegloba l.com Internet Source	<1 %
21	Jian Zheng, Wei Yang, Changchun Wang, Dandan Jiang, Dan Wang, Qi Yang, Yijiao Zhang. "Chapter 4 Research on Complaint Sensitivity Analysis Based on Random Forest Algorithm", Springer Science and Business Media LLC, 2023 Publication	<1 %
22	discuss.streamlit.io Internet Source	<1 %
23	www.mdpi.com Internet Source	<1 %
24	sebsauvage.net Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 About the company:

BDB (Big Data BizViz) is a data analytics firm that was founded in 2015 by Avin Jain. The company has developed its own BI (Business Intelligence) tool that is based on a microservices architecture. This modern platform can be easily integrated with popular business applications or customized to create specific analytics flows.

The BDB tool incorporates elements of augmented analytics, machine learning, and top-end visualization to provide a comprehensive data analytics solution. It is widely used by all layers of an organization including CXOs, Citizen Data Scientists, Data Scientists, Business Analysts, and Business Users, to gain insights and make informed decisions based on data-driven analytics.

With BDB's platform, users can easily visualize complex data sets and identify patterns and trends that can help them optimize their business processes. Overall, BDB offers a powerful data analytics solution that helps organizations make informed decisions, improve performance, and stay competitive in today's data-driven business environment..

1.2 Projects of the Company:

As a data analytics firm, BDB (Big Data BizViz) has worked on a number of projects for clients in various industries. Here are some examples of projects that BDB has worked on:

Healthcare Analytics: BDB has worked on a project with a large healthcare organization to analyze patient data and identify patterns that can help improve patient outcomes. BDB's platform was used to visualize and analyze data from electronic health records, medical devices, and other sources.

Fraud Detection: BDB has worked on a project with a financial services company to develop a fraud detection system. BDB's platform was used to analyze large volumes of transactional data and identify anomalies that could indicate fraudulent activity.

Supply Chain Optimization: BDB has worked on a project with a manufacturing company to optimize its supply chain operations. BDB's platform was used to analyze data from various sources including suppliers, inventory, and logistics, to identify opportunities for improvement and cost savings.

Customer Analytics: BDB has worked on a project with a retail company to analyze customer data and identify trends in buying behavior. BDB's platform was used to visualize data from point-of-sale systems, customer loyalty programs, and other sources to identify patterns and develop targeted marketing campaigns.

Overall, BDB has worked on a range of projects using its platform to help clients gain insights from their data and optimize their business processes.

1.3 Services Provided:

BDB (Big Data BizViz) provides a range of data analytics services to help organizations make informed decisions based on their data. Here are some of the services that BDB provides:

- Business Intelligence (BI) Consulting: BDB offers BI consulting services to help organizations identify their data needs and implement a customized analytics solution that meets their specific requirements.

- Data Integration: BDB helps organizations integrate data from multiple sources into a single analytics platform, making it easier to analyze and gain insights from their data.

- Data Visualization: BDB provides data visualization services to help organizations easily view and understand their data. The platform provides top-end visualization techniques like augmented analytics, interactive dashboards, and machine learning to provide actionable insights.

- Predictive Analytics: BDB leverages machine learning algorithms to provide predictive analytics services that help organizations forecast future trends and identify potential risks and opportunities.

- Data Governance and Security: BDB provides services to ensure data governance and security, enabling organizations to maintain compliance and protect their sensitive data.

Overall, BDB provides a comprehensive data analytics solution that helps organizations to leverage their data assets to improve performance, reduce costs, and make informed decisions.

2. ABOUT THE PROJECT

2.1Project Title:

Electricity Generation Analysis using BDB tool

2.2 Introduction

Electricity generation analysis is a critical aspect of the energy industry that involves evaluating the performance and efficiency of power plants. With increasing global demand for electricity, the need for efficient and reliable electricity generation has become more crucial than ever. As a result, there is a growing interest in using data analytics tools and techniques to optimize electricity generation and reduce costs.

Electricity generation analysis involves gathering data on power generation, consumption, and distribution, and using this data to evaluate the efficiency of power plants. The data collected

may include factors such as the amount of fuel consumed, the amount of electricity generated, and the cost of production. The analysis can be done in real-time, providing power plant operators with immediate insights into their operations and allowing them to make informed decisions to optimize their processes.

Data analytics tools such as the BDB (Balance of Data) tool can be used to monitor and analyze the performance of power plants in real-time, providing insights into areas of inefficiency and identifying opportunities for optimization. By identifying areas of inefficiency, power plants can implement changes to their operations that can result in significant cost savings and improved performance. Additionally, electricity generation analysis can help power plants reduce their environmental impact by identifying areas where energy is being wasted or by optimizing the use of renewable energy sources.

In summary, electricity generation analysis is a critical aspect of the energy industry that can help power plants optimize their resources, improve efficiency, reduce costs, and reduce their environmental impact. By leveraging data analytics tools and techniques, power plants can make informed decisions and remain competitive in a rapidly evolving energy landscape.

2.3 Technology Used:

- Big Data Analytics: Big data analytics tools are used to process and analyze the large amounts of data collected from power plants. These tools can identify patterns, trends, and areas of inefficiency, allowing power plant operators to optimize their operations.

- Machine Learning: Machine learning algorithms are used to identify patterns and predict future outcomes based on historical data. In electricity generation analysis, machine learning can be used to predict energy consumption and production, identify potential issues before they occur, and optimize power plant operations.

- Visualization Tools: Visualization tools are used to present data in a way that is easy to understand, allowing power plant operators to quickly identify areas of inefficiency and opportunities for optimization.

2.4 Industrial Scope

Electricity generation analysis has a significant industrial scope, as it is relevant to a range of industries and sectors that rely on electricity for their operations. Here are some examples of industries where electricity generation analysis is particularly relevant:

Power Generation: The power generation industry is one of the most significant areas where electricity generation analysis is used. Power plants use data analytics tools to optimize their operations, improve efficiency, and reduce costs.

Manufacturing: The manufacturing industry relies heavily on electricity for its operations, and electricity generation analysis can help optimize energy usage and reduce costs. By identifying areas where energy is being wasted, manufacturers can make changes to their operations that can result in significant savings.

Transportation: The transportation industry, particularly the electric vehicle segment, is

experiencing significant growth, and electricity generation analysis can help optimize the use of electric vehicle charging stations, reducing costs and improving the efficiency of the charging process.

Healthcare: The healthcare industry relies heavily on electricity for its operations, particularly in hospital settings. Electricity generation analysis can help hospitals optimize energy usage, reducing costs and improving the sustainability of their operations.

Agriculture: The agriculture industry is also increasingly reliant on electricity for its operations, particularly in areas such as irrigation and crop processing. Electricity generation analysis can help farmers optimize energy usage, reducing costs and improving the efficiency of their operations.

In summary, electricity generation analysis has a broad industrial scope, with applications in a range of industries and sectors that rely on electricity for their operations. By leveraging data analytics tools and techniques, these industries can optimize their energy usage, reduce costs, and improve their environmental sustainability.

2.5 Goal

The goal of electricity generation analysis is to optimize the performance and efficiency of power plants by using data analytics tools and techniques. By analyzing data related to power generation, consumption, and distribution, electricity generation analysis can help identify areas of inefficiency and opportunities for optimization. The ultimate goal is to improve the overall performance of power plants, reduce costs, and increase their environmental sustainability.

3. Project Work Flow

3.1 Problem Statement

The need to optimize the performance, efficiency, and cost-effectiveness of power generation processes. Despite significant advancements in technology, power plants still face challenges in maintaining optimal performance and efficiency. Some of the challenges include:
Energy Waste: Power plants may waste energy due to factors such as equipment malfunctions, inefficient processes, or inadequate maintenance.
Environmental Impact: Power generation processes can have a significant environmental impact, particularly with regard to greenhouse gas emissions.
Cost: Power plants face significant cost pressures, with fuel costs, maintenance costs, and labor

costs all contributing to overall operating expenses.
To address these challenges, power plants need to leverage data analytics tools and techniques to optimize their operations and reduce costs. The use of data analytics can help identify areas of inefficiency, allowing power plants to make informed decisions to improve their processes and reduce energy waste. Additionally, data analytics can help power plants optimize the use of renewable energy sources, reducing their environmental impact. Overall, the problem statement for electricity generation analysis is the need to optimize power generation processes to improve efficiency, reduce costs, and minimize environmental impact.

4. Data Extraction and Preparation

4.1 Request API from EIA

Navigate to EIA (Environment Impact Analysis)
We extracted data from the EIA (Environment Impact Analysis) website, which provides hourly electricity generation data for various plants in the USA.
This website provides data about electricity generated by various plants in USA on hourly basis

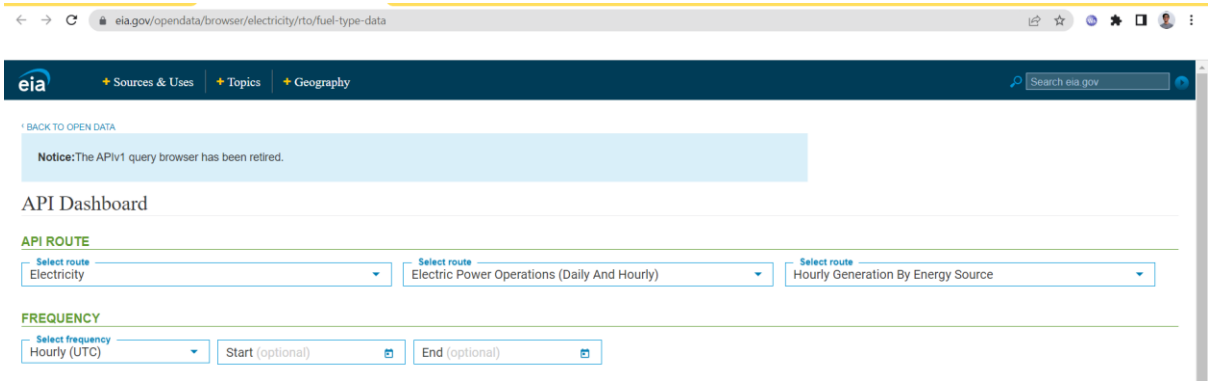


Fig.4.1 API Dashboard

4.2 Data Scraping

The data extraction was performed using a Python script that utilized the EIA API. The extracted data was cleaned, processed, and stored in a database for further analysis.
Code:

```
import requests
import datetime
import pandas as pd
from 4.2 Data import Label Encoder

def data extract():

    api_key = 'lS07rlQD1KkhGcZ9BMVfexEKONjRpbYtUYJodOR8'
    #today = datetime.date.today() - datetime.timedelta(days=4)
    url1 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start=2023-04-
11T00&end=2023-04-
11T12&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'
    url2 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start=2023-04-
11T13&end=2023-04-
11T23&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'
    url3 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start=2023-04-
```

```
12T00&end=2023-04-
12T12&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'
    url4 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start=2023-04-
12T13&end=2023-04-
12T23&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'
    url5 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start=2023-04-
13T00&end=2023-04-
13T12&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'
    url6 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start=2023-04-
13T13&end=2023-04-
13T23&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'
    url7 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start=2023-04-
14T00&end=2023-04-
14T12&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'
    url8 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start=2023-04-
14T13&end=2023-04-
14T23&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'

    le = LabelEncoder()
    rows = []
    for i in [1,2,3,4,5,6,7,8]:
        url = eval(f'url{i}')
        response = requests.get(url)

        if response.status_code == 200:
            data = response.json()['response']['data']
            for d in data:
                date = d['period'] + '-01'  # add day to format as YYYY-MM-DD
                Respondent_name = d['respondent-name']
                Type_name = d['type-name']
                Value = d['value']
                Value_unit = d['value-units']
                rows.append({'YearMonth': date, 'respondent_name': Respondent_name,
'type_name': Type_name, 'value': Value, 'value_units': Value_unit})
        else:
            print(f'Request failed with status code {response.status_code}.')

    if len(rows) == 0:
        print('No data available.')
        return pd.DataFrame()

    df = pd.DataFrame(rows)
    df = df.dropna()
    df['YearMonth'] = pd.to_datetime(df['YearMonth'], format='%Y-%m-%dT%H-%M')
    df['date']=df['YearMonth'].dt.strftime('%Y-%m-%d')
```

```
    df['time']=df['YearMonth'].dt.strftime('%H:%M')
    df = df.drop('YearMonth', axis=1)
    df['type_name_id'] = le.fit_transform(df['type_name'])
    df['respondent_name_id'] = le.fit_transform(df['respondent_name'])
    return df
```

modified code to extract code on daily basis:
code:

```
import requests
import datetime
import pandas as pd
import clickhouse_driver

def data_extract():

    api_key = 'lS07rlQD1KkhGcZ9BMVfexEKONjRpbYtUYJodOR8'
    today = datetime.date.today() - datetime.timedelta(days=2)

    url1 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start={today}T00&end={
today}T12&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'
    url2 = f'https://api.eia.gov/v2/electricity/rto/fuel-type-
data/data/?api_key={api_key}&frequency=hourly&data[0]=value&start={today}T13&end={
today}T23&sort[0][column]=period&sort[0][direction]=asc&offset=0&length=5000'

    clickhouse_host = 'clickhouse.clickhouse'
    clickhouse_port = 9000
    clickhouse_user = 'darren_andrew'
    clickhouse_database='db_darren_andrew'
    clickhouse_password = 'Tg4246'

    connection = clickhouse_driver.connect(
        host=clickhouse_host,
        port=clickhouse_port,
        user=clickhouse_user,
        database=clickhouse_database,
        password=clickhouse_password
    )

    cursor = connection.cursor()

    # Query ClickHouse tables to retrieve labels
    cursor.execute('SELECT respondent_name, respondent_name_id FROM
Electricity_respondent_id')
    respondent_labels = cursor.fetchall()

    cursor.execute('SELECT type_name, type_name_id FROM Electricity_type_id')
    type_labels = cursor.fetchall()

    rows = []
```

```
    for i in [1, 2]:
        url = eval(f'url{i}')
        response = requests.get(url)

        if response.status_code == 200:
            data = response.json()['response']['data']
            for d in data:
                date = d['period'] + '-01'  # add day to format as YYYY-MM-DD
                respondent_name = d['respondent-name']
                type_name = d['type-name']
                value = d['value']
                value_unit = d['value-units']
                rows.append({'YearMonth': date, 'respondent_name': respondent_name,
'type_name': type_name, 'value': value, 'value_units': value_unit})
        else:
            print(f'Request failed with status code {response.status_code}.')

    if len(rows) == 0:
        print('No data available.')
        return pd.DataFrame()

    df = pd.DataFrame(rows)
    df = df.dropna()
    df['YearMonth'] = pd.to_datetime(df['YearMonth'], format='%Y-%m-%dT%H-%M')
    df['date'] = df['YearMonth'].dt.strftime('%Y-%m-%d')
    df['time'] = df['YearMonth'].dt.strftime('%H:%M')
    df = df.drop('YearMonth', axis=1)

    # Replace label encoding with ClickHouse labels
    respondent_labels_dict = dict(respondent_labels)
    df['respondent_name_id'] = df['respondent_name'].map(respondent_labels_dict)
    type_labels_dict=dict(type_labels)
    df['type_name_id']=df['type_name'].map(type_labels_dict)
    return df
```

4.3 Deploy it into pipeline

the above code is written in python script component in Pipeline editor of BDB tool

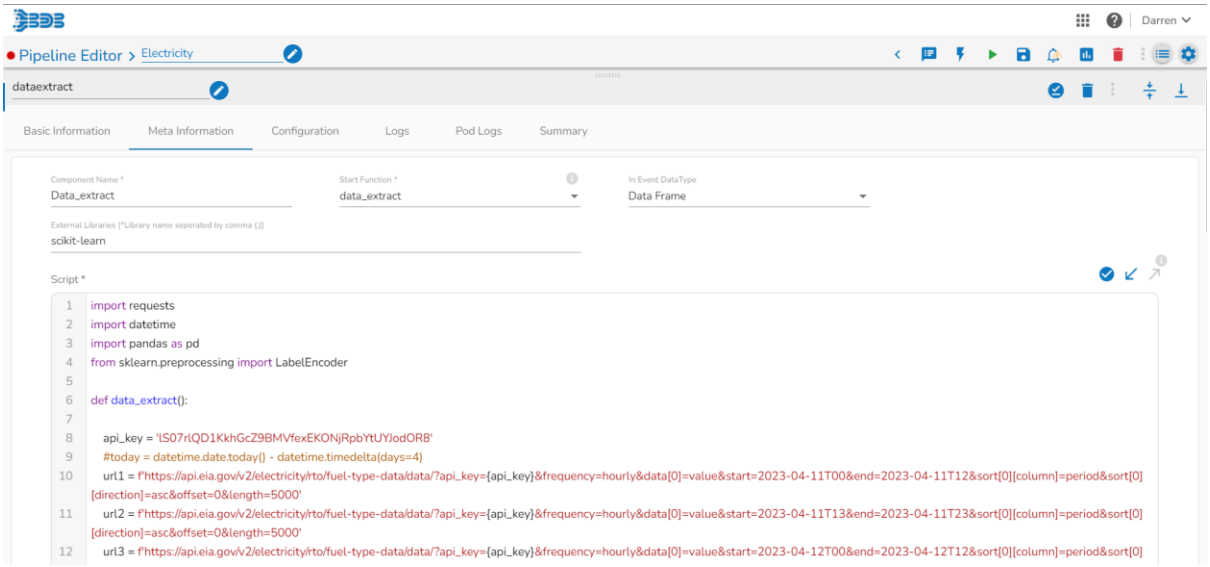


Fig.4.3.1 Python component(Data extract code)

To write the scrapped data into an Data Base , add a DB writer to the pipeline editor .

Before adding the DB writer we need add an Kafka event in between the python script component and DB writer
The output from the python script component is reflected in the kafka event below

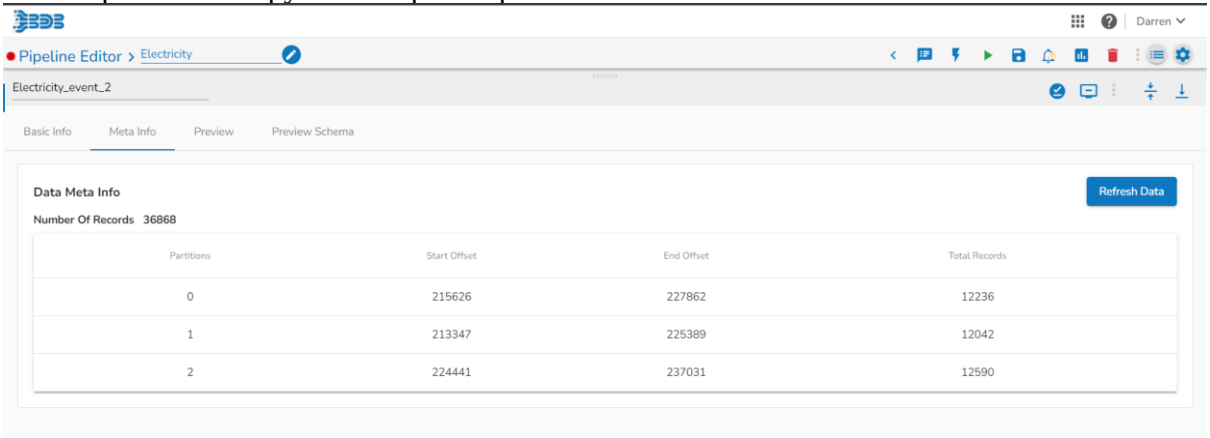
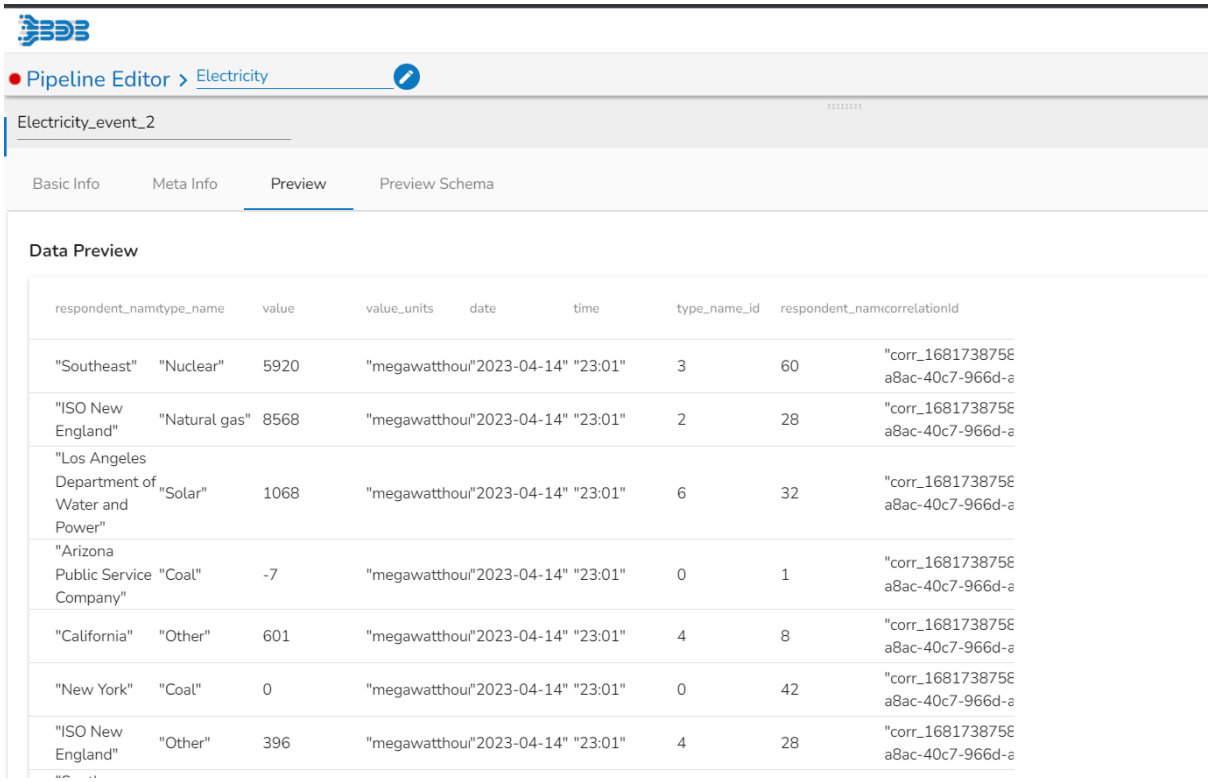


Fig.4.3.2 Kafka event (Data extract code)

From the above image we can infer that the python component has sent 36868 records from eia.
Here is a preview of the data depicted in the image below.



respondent_nam	type_name	value	value_units	date	time	type_name_id	respondent_nam	correlationId
"Southeast"	"Nuclear"	5920	"megawatthou	"2023-04-14"	"23:01"	3	60	"corr_168173875E a8ac-40c7-966d-a
"ISO New England"	"Natural gas"	8568	"megawatthou	"2023-04-14"	"23:01"	2	28	"corr_168173875E a8ac-40c7-966d-a
"Los Angeles Department of Water and Power"	"Solar"	1068	"megawatthou	"2023-04-14"	"23:01"	6	32	"corr_168173875E a8ac-40c7-966d-a
"Arizona Public Service Company"	"Coal"	-7	"megawatthou	"2023-04-14"	"23:01"	0	1	"corr_168173875E a8ac-40c7-966d-a
"California"	"Other"	601	"megawatthou	"2023-04-14"	"23:01"	4	8	"corr_168173875E a8ac-40c7-966d-a
"New York"	"Coal"	0	"megawatthou	"2023-04-14"	"23:01"	0	42	"corr_168173875E a8ac-40c7-966d-a
"ISO New England"	"Other"	396	"megawatthou	"2023-04-14"	"23:01"	4	28	"corr_168173875E a8ac-40c7-966d-a

Fig 4.3.3 Preview of Extract data

The data is then written to a database with the help of the DB writer by providing the necessary credentials

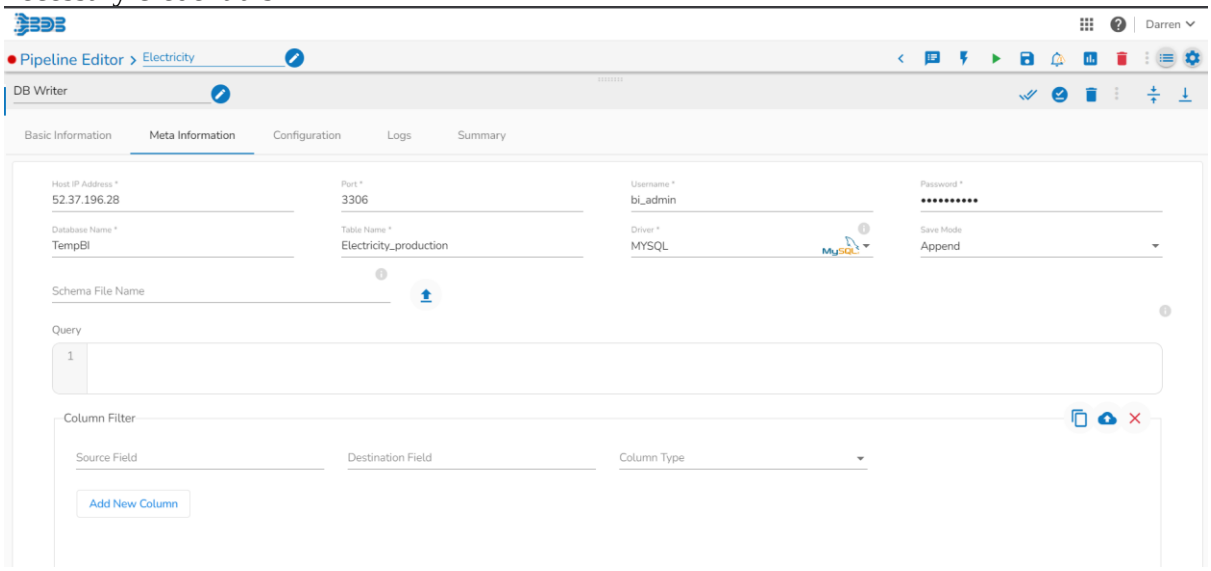


Fig 4.3.4 Configuration of Database

The below image shows that the data is stored in DataBase

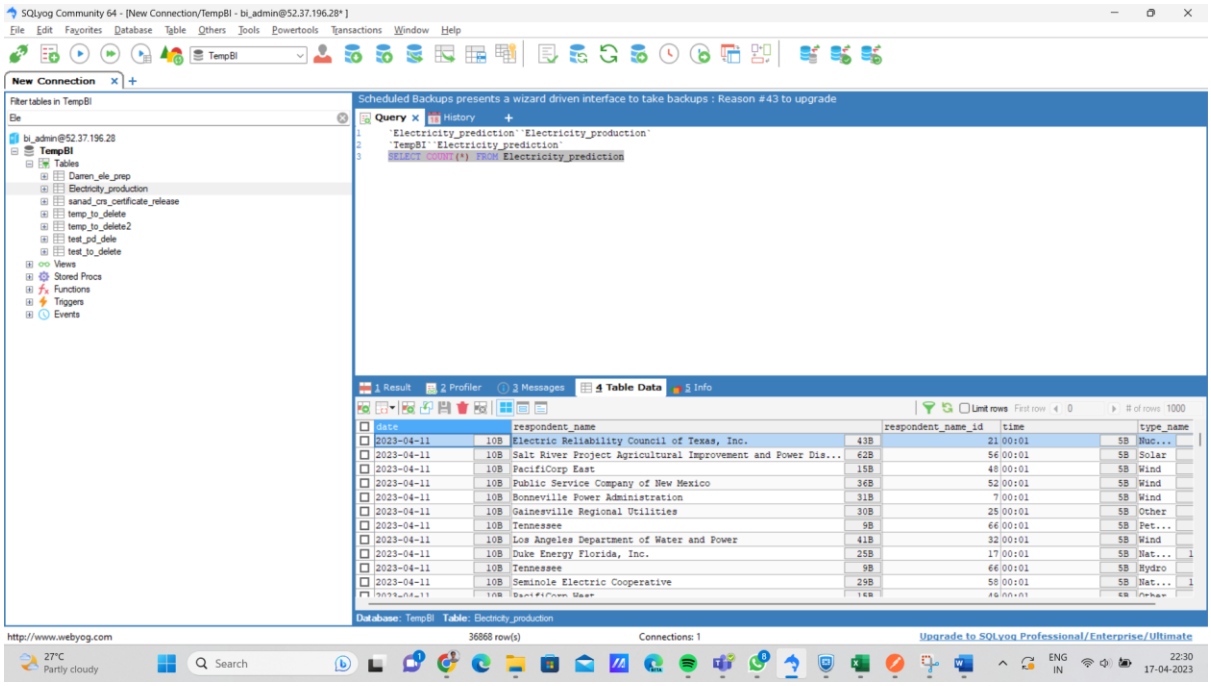


Fig 4.3.5 data is stored in DataBase

5. Training the model

Let's use this data to train the regression model:

This part of the workflow involves training a model using the above data.

5.1 Connect to the database

At first we need to connect the Database using the Data connector option from data centre module.

Here I'm selecting mysql as my dataconnector for this project.

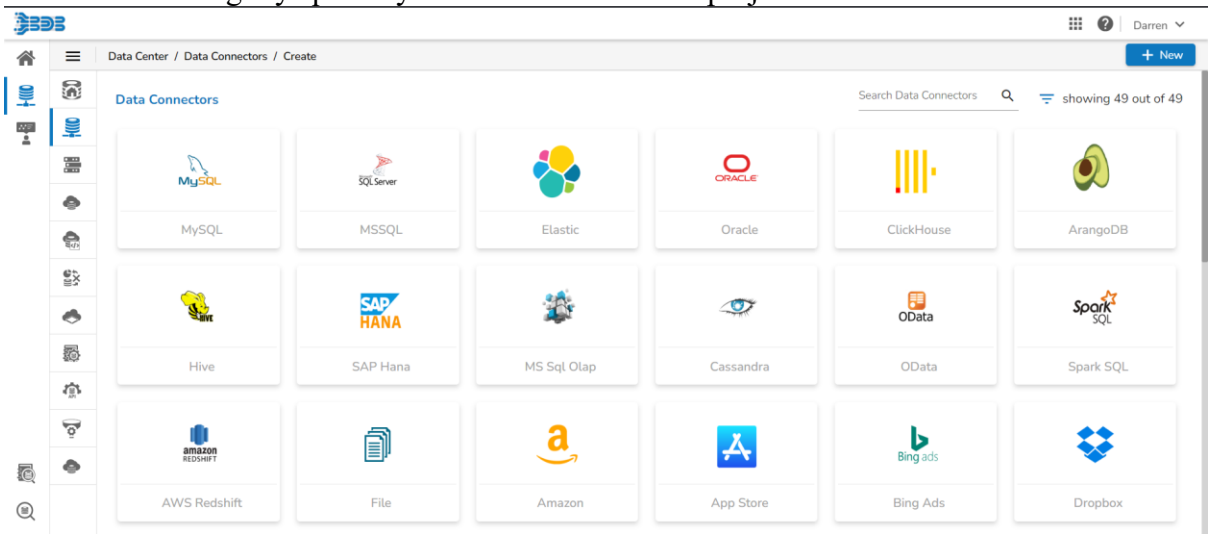


Fig.5.1.1 DataConnectors

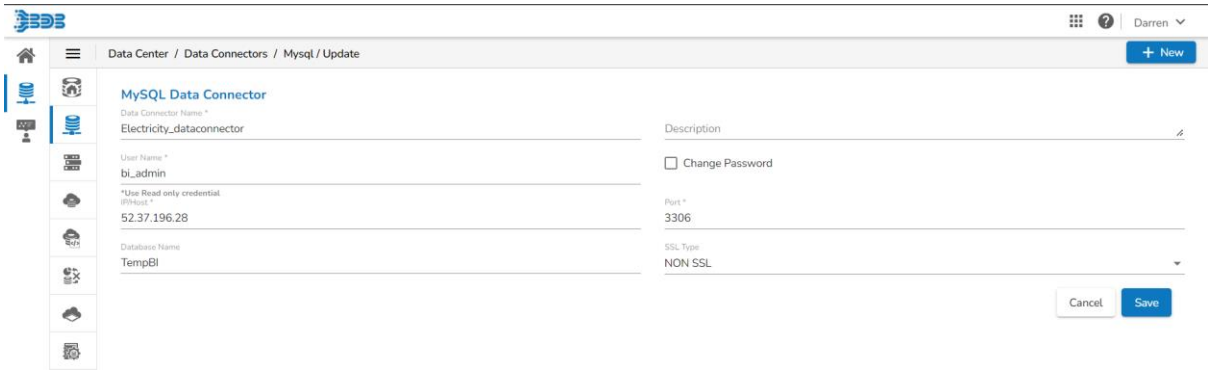


Fig.5.1.2 configuring DataConnector

Now let's create a new dataset from the newly created dataconnector.

The Below image is a view of dataset page. The following sql query is given to retrieve the data.

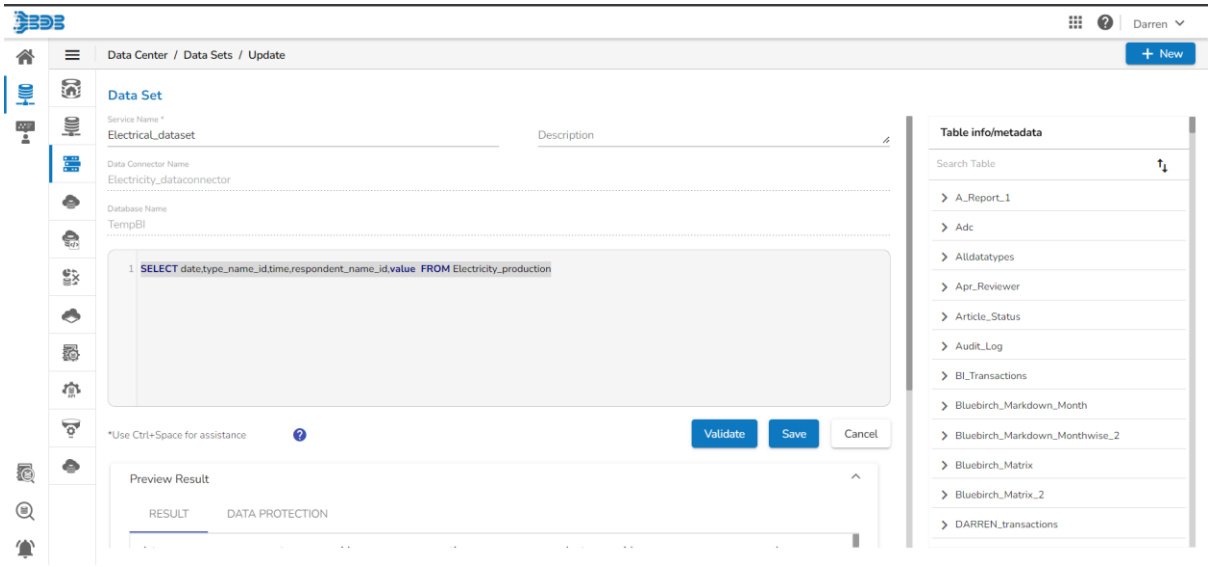
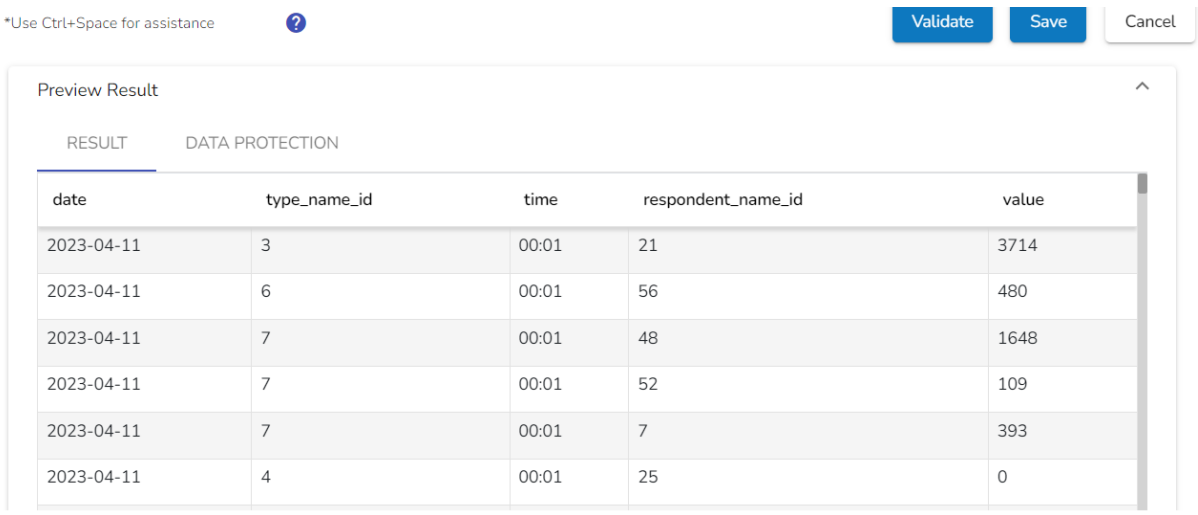


Fig.5.1.3 Configuring dataset



date	type_name_id	time	respondent_name_id	value
2023-04-11	3	00:01	21	3714
2023-04-11	6	00:01	56	480
2023-04-11	7	00:01	48	1648
2023-04-11	7	00:01	52	109
2023-04-11	7	00:01	7	393
2023-04-11	4	00:01	25	0

Fig.5.1.4 preview of data

Validate the above query and click on save.
From here we need to move to DS Labs. At first we need to create a project and activate it.

5.2 Upload the Dataset toAutoML

At first create a project in DS labs and then Activate it.

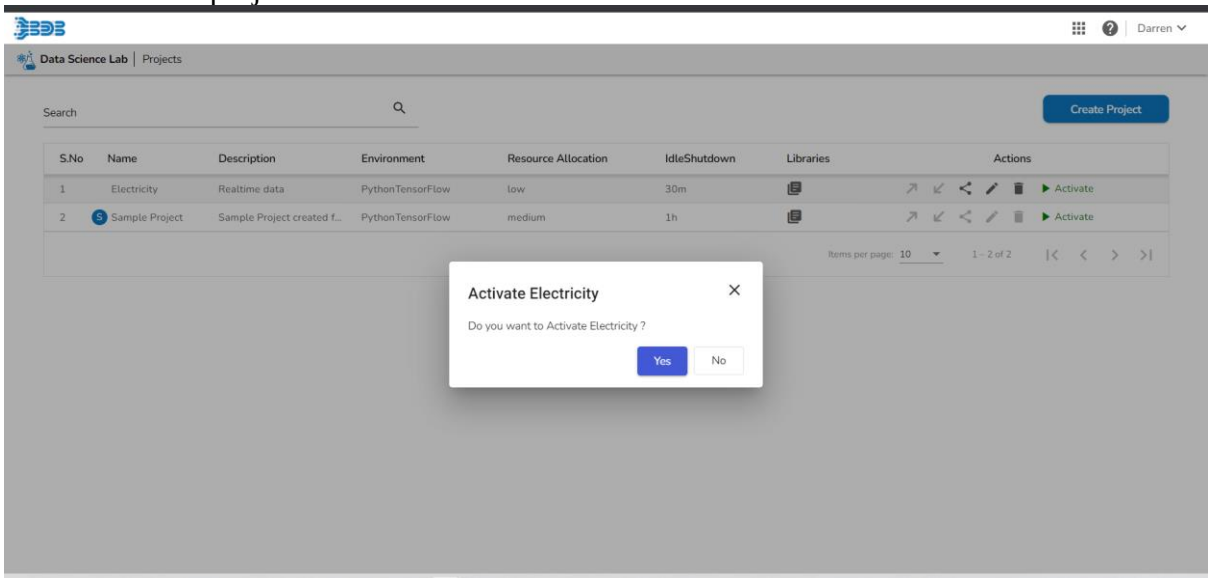


Fig.5.2.1 Configuring DS labs

Click on the dataset tab

Click on add dataset. Select the dataset that was created using the above steps.
Once the dataset is added to the DS Lab click on create experiment.

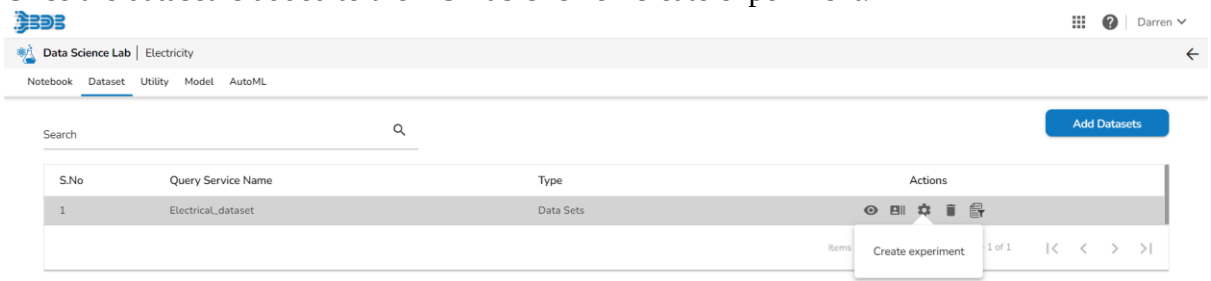


Fig.5.2.2 Creating AutoML Experiment

It will take you to a page to configure the automl experiment. Give a name and the target

column (Here I'm selecting value as my target column).

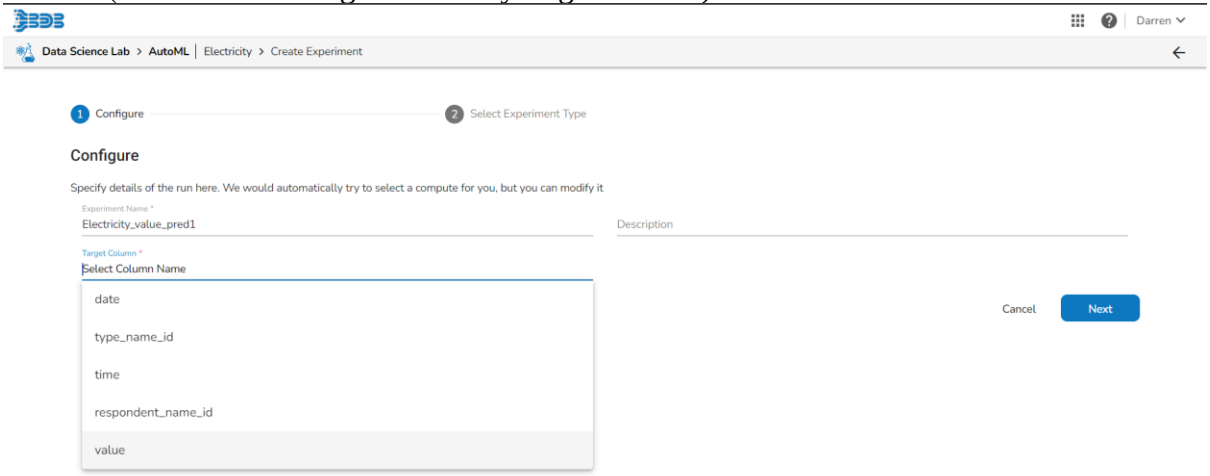


Fig.5.2.3 Configuring AutoML

Click on next. In this page we have an option to select classification or regression (Here I will be selecting regression.)

From here the experiment will be created and after few minutes the automl will complete its execution (as seen from the below image status==completed).

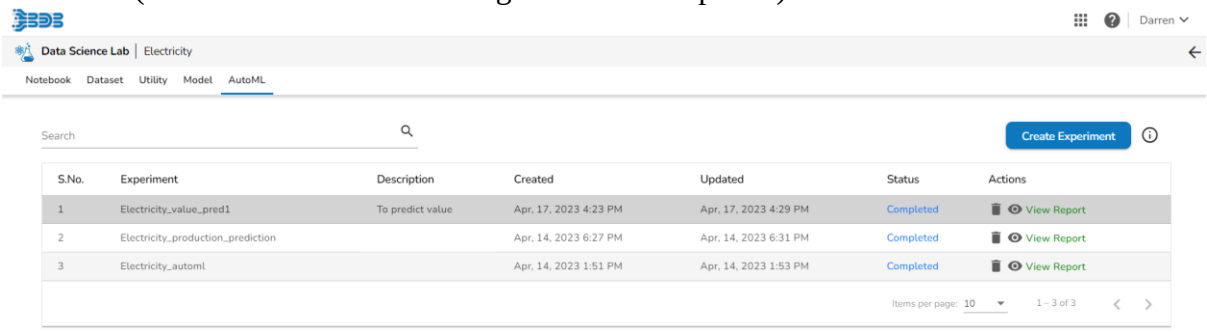


Fig.5.2.4 Created AutoML experiment

The automl feature in DS Labs recommends the best model for the given dataset. The below image gives a brief explaination it.

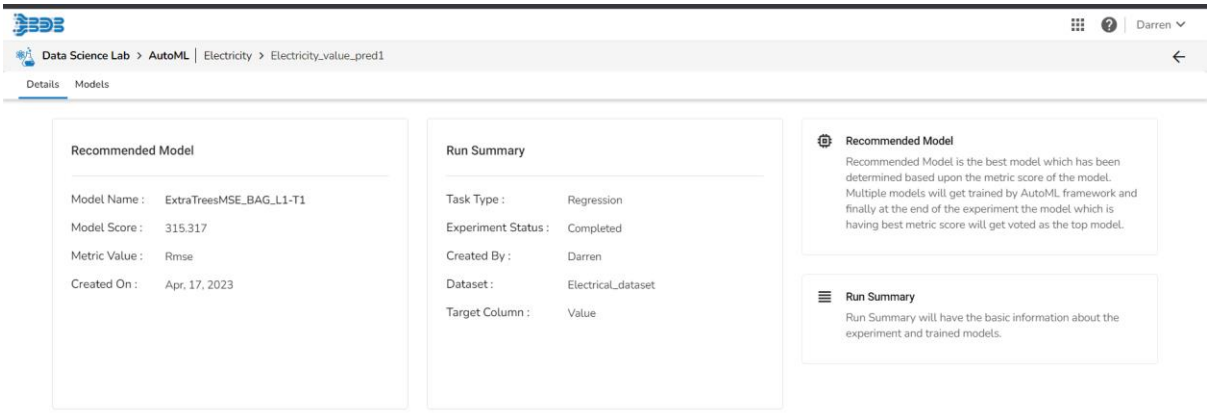


Fig.5.2.5 Recommended model

Moreover the automl suggests a total of 3 models which could be suitable for the dataset with its rmse score

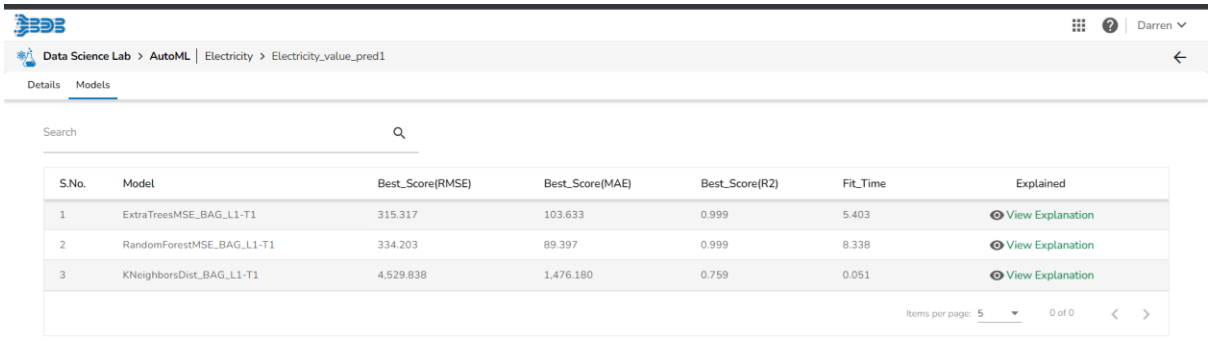


Fig.5.2.6 List of Models

We can also get information of this model starting with feature importance, Regression stats, Feature dependence and so on.

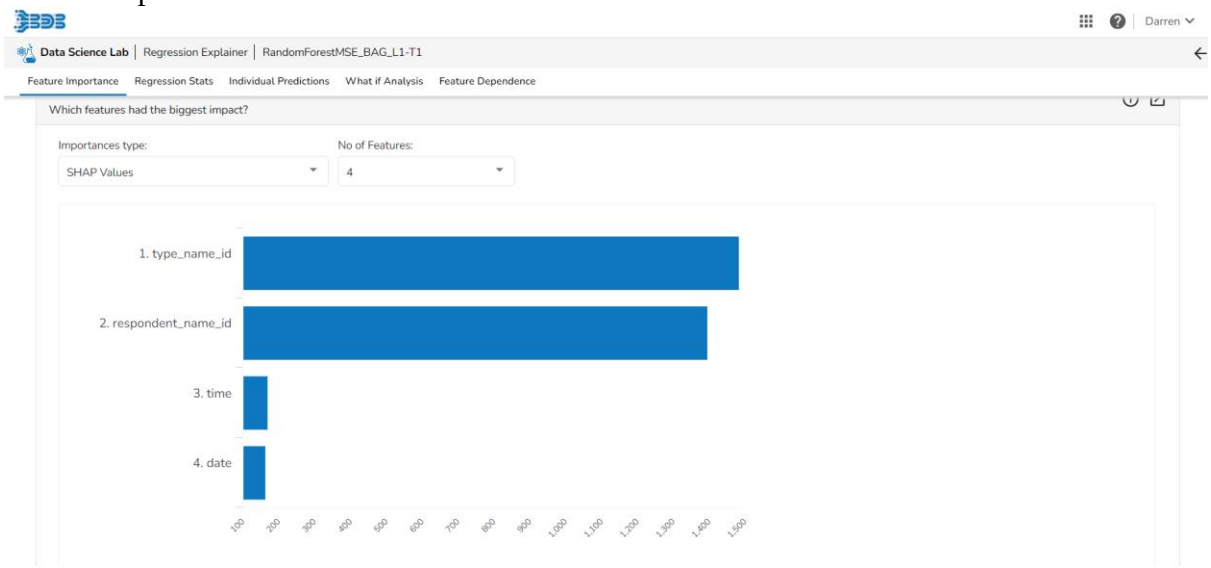


Fig.5.2.7 Model explainer

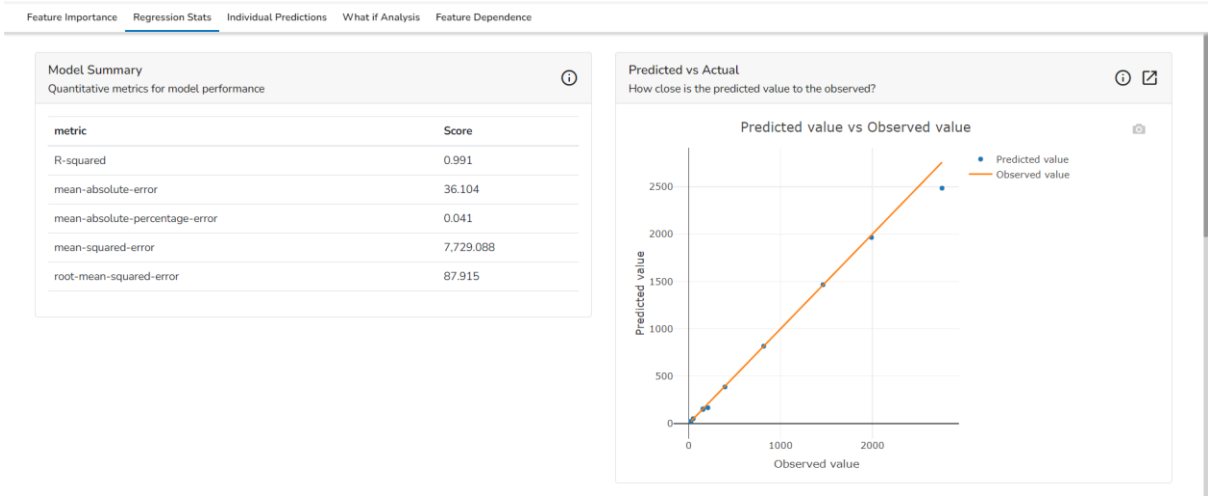


Fig.5.2.8 Model summary

From here the model needs to be uploaded to the pipeline.
Navigate to the Model tab change the filters to all from registered.
From here we could see the automl experiment.
Expand the models and click on the register button.

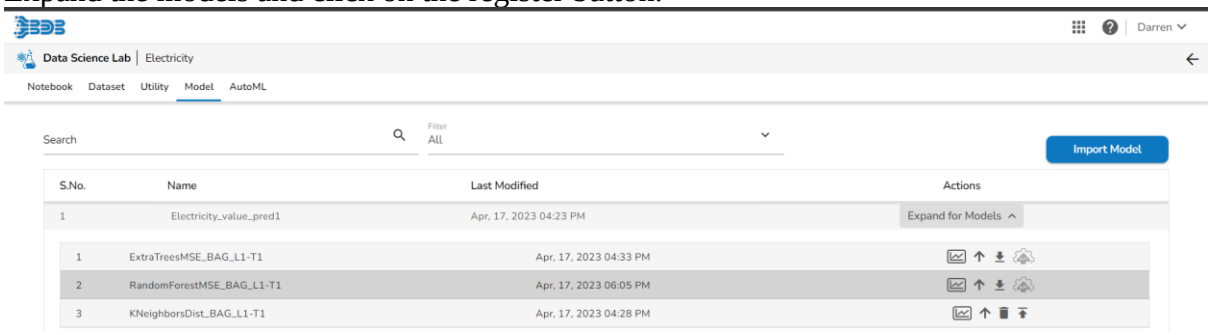


Fig.5.2.9 register model

5.3 Upload the model to pipeline

In the pipeline drag and drop the automl component
Configure the component by selecting the DS Labproject and the registered model

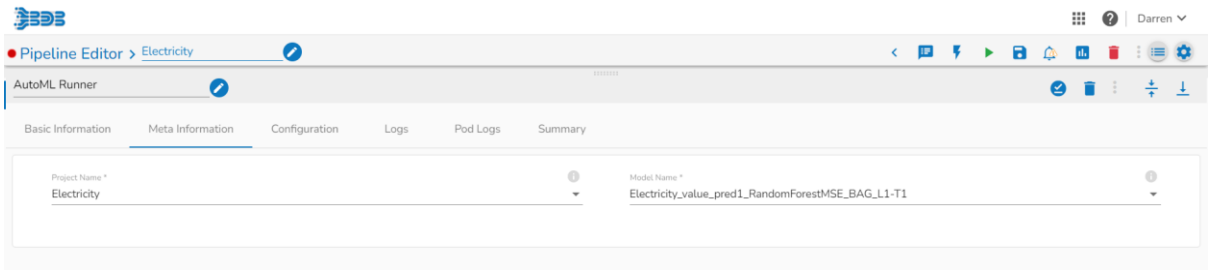


Fig 5.3.1 Configure AutoML component

Here is a preview of the data depicted in the image below

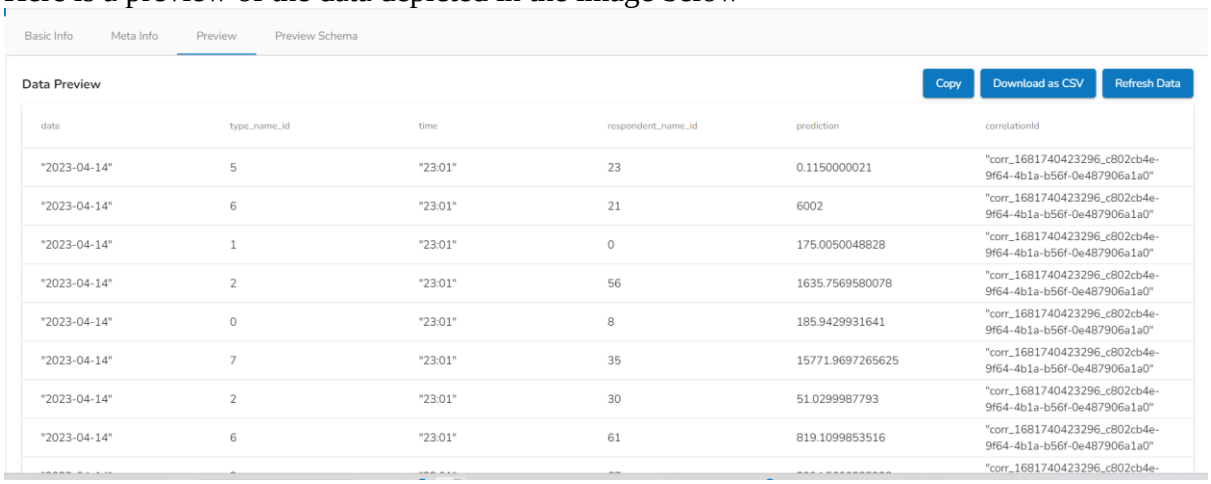


Fig.5.3.2 Output of the model

As we can see in the above image that the AutoML component has added a new column “predictions” where all the predicted values are stored

The above data is then stored in another DB writer

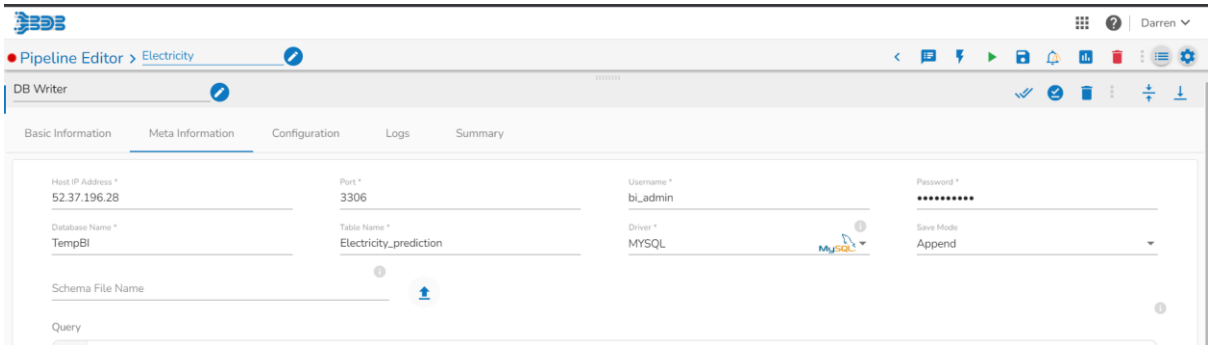


Fig.5.3.3 Configuring DB writer

6. Prepare dataset for next 6 days:

Once the model is trained, we can create a dataset to be input into the AutoML component, which will generate predicted values as the output.

Code:

```
import pandas as pd
from datetime import date, timedelta

def replicate_dataframe_with_dates(df):
    # Get yesterday's date
    yesterday = date.today() - timedelta(days=1)

    # Copy the original DataFrame multiple times with different dates starting from yesterday
    new_dates = pd.date_range(start=yesterday, periods=6, freq='D')
    copied_dfs = []

    for new_date in new_dates:
        copied_df = df.copy()
        copied_df['date'] = new_date.strftime('%Y-%m-%d')
        copied_dfs.append(copied_df)

    # Concatenate the modified DataFrames with the original DataFrame
    combined_df = pd.concat([df] + copied_dfs, ignore_index=True)

    # Drop unnecessary columns and reorder columns
    combined_df = combined_df.drop(['type_name', 'respondent_name', 'value_units', 'value'],
axis=1)
    combined_df = combined_df[['date', 'type_name_id', 'time', 'respondent_name_id']]

    return combined_df
```

The output generated from the previous code is subsequently fed into the AutoML component within the pipeline. The resulting output of the AutoML component contains the predicted value. Afterwards, the data is transmitted to another Python script. The following script prepares the data for visualization purposes before storing it in a database.

Code:

```
import requests
```

```
import datetime
import pandas as pd
import clickhouse_driver

def data_combine(df):


    clickhouse_host = 'clickhouse.clickhouse'
    clickhouse_port = 9000
    clickhouse_user = 'darren_andrew'
    clickhouse_database='db_darren_andrew'
    clickhouse_password = 'Tg4246'

    connection = clickhouse_driver.connect(
        host=clickhouse_host,
        port=clickhouse_port,
        user=clickhouse_user,
        database=clickhouse_database,
        password=clickhouse_password
    )

    cursor = connection.cursor()

    # Query ClickHouse tables to retrieve labels
    cursor.execute('SELECT respondent_name_id, respondent_name FROM
Electricity_respondent_id')
    respondent_labels = cursor.fetchall()

    cursor.execute('SELECT type_name_id, type_name FROM Electricity_type_id')
    type_labels = cursor.fetchall()

    # Replace label encoding with ClickHouse labels
    respondent_labels_dict = dict(respondent_labels)
    df['respondent_name'] = df['respondent_name_id'].map(respondent_labels_dict)
    type_labels_dict=dict(type_labels)
    df['type_name']=df['type_name_id'].map(type_labels_dict)
    return df
```

The above data is stored in a Database.

The below picture depicts the complete pipeline.

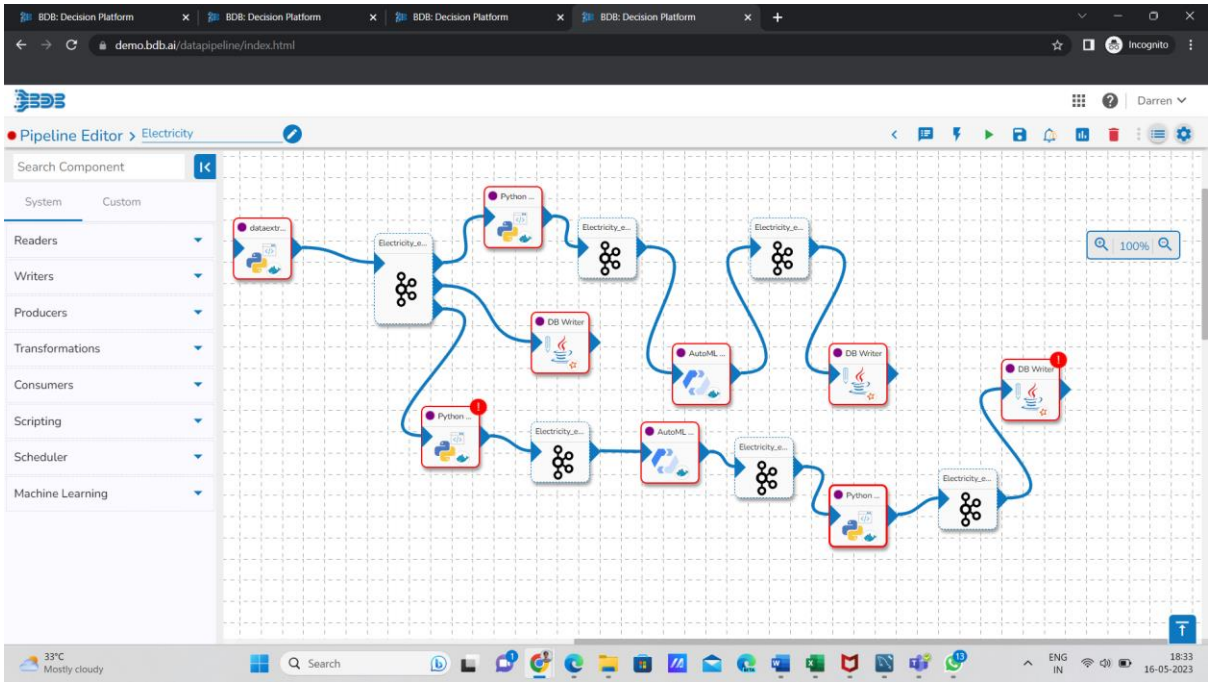


Fig.6 Pipeline editor

7. Forecasting:

Electricity generation plays a crucial role in ensuring the stability and efficiency of power systems. Accurate forecasting of electricity generation can aid in efficient load management, resource planning, and decision-making processes. In this project, we aim to develop an ESM forecast model to predict electricity generation for the next 24 hours.

For this project, we retrieved data from a ClickHouse database. The relevant columns included the date, time, and electricity generation values. To ensure consistency and handle missing values, we resampled the data to an hourly frequency. Additionally, we performed linear interpolation and backward filling to handle missing values and outliers.

Exponential Smoothing (ESM) was chosen as the forecasting technique for this project. ESM is a time series forecasting method that captures trend, seasonality, and smoothing levels. To determine the best model parameters, we conducted cross-validation using TimeSeriesSplit. The evaluation metric used was the root mean squared error (RMSE), which provides a measure of the model's accuracy.

Code:

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import TimeSeriesSplit
from clickhouse_driver import Client

def predict_time_series():
    client = Client(host='clickhouse.clickhouse', port='9000', database='db_darren_andrew', user='darren_andrew', password='Tg4246')
```

```
    # Execute query
    query = "SELECT * FROM Electricity_production"
    results = client.execute(query)
    df = pd.DataFrame(results, columns=['date', 'respondent_name',
'respondent_name_id','time','type_name','type_name_id','value','value_units'])
    # Close database connection
    client.disconnect()
    df['timestamp'] = pd.to_datetime(df['date'] + ' ' + df['time'])
    df = df.set_index('timestamp')

    # resample data to hourly frequency
    df = df.resample('H').sum()

    # handle missing values and outliers
    df = df.replace(0, np.nan)
    df = df.interpolate(method='linear', limit_direction='both')
    df = df.fillna(method='bfill')
    df['timestamp'] = df.index
    df['date'] = df['timestamp'].dt.strftime('%Y-%m-%d')
    df['time'] = df['timestamp'].dt.strftime('%H:%M')
    df['timestamp'] = df['timestamp'].dt.strftime('%Y-%m-%d %H:%M')


    # split data into training and testing sets
    split_index = len(df) - 24
    train_data = df.iloc[:split_index]
    test_data = df.iloc[split_index:]

    # define parameter grid for exponential smoothing
    alpha = np.arange(0.1, 1.0, 0.1)
    beta = np.arange(0.1, 1.0, 0.1)
    gamma = np.arange(0.1, 1.0, 0.1)

    # perform cross-validation to find best model
    best_score, best_params = float("inf"), None
    cv = TimeSeriesSplit(n_splits=5)
    for train_index, val_index in cv.split(train_data):
        train = train_data.iloc[train_index]
        val = train_data.iloc[val_index]
        for a in alpha:
            for b in beta:
                for g in gamma:
                    try:
                        model = ExponentialSmoothing(train['value'], trend='add', seasonal='add',
seasonal_periods=24).fit(smoothing_level=a, smoothing_slope=b, smoothing_seasonal=g)
                        y_pred = model.forecast(steps=len(val))
                        score = np.sqrt(mean_squared_error(val['value'], y_pred))
                        if score < best_score:
                            best_score, best_params = score, (a, b, g)
                            mape = np.mean(np.abs((val['value'] - y_pred) / val['value'])) * 100
```

```
                # calculate the accuracy as a percentage
                accuracy = 100 - mape
            except:
                continue

    # fit the best model on the entire dataset
    forecast = ExponentialSmoothing(df['value'], trend='add', seasonal='add',
seasonal_periods=24)
    model = forecast.fit(smoothing_level=best_params[0], smoothing_slope=best_params[1],
smoothing_seasonal=best_params[2])

    # make predictions for the next 24 hours after the available data
    next_24_hours = pd.date_range(start=df.index[-1] + pd.DateOffset(hours=1), periods=24,
freq='H')
    next_24_hours = pd.DatetimeIndex(next_24_hours)
    next_24_hours_df = pd.DataFrame(index=next_24_hours)
    next_24_hours_df['date']=next_24_hours_df.index.strftime('%Y-%m-%d')
    next_24_hours_df['time']=next_24_hours_df.index.strftime('%H:%M')
    next_24_hours_df['timestamp']=next_24_hours_df.index.strftime('%Y-%m-%d %H:%M')
    next_24_hours_predictions = model.forecast(steps=24)
    pred_df = pd.concat([df, next_24_hours_df])
    next_24_hours_df['test_predictions']=next_24_hours_predictions


    return next_24_hours_df
```

The output gives forecast for next 24 hrs

The dataset was split into training and testing sets, with the last 24 hours reserved for testing. Through cross-validation, we identified the best model parameters that yielded the lowest RMSE. The chosen model demonstrated excellent performance, capturing the underlying patterns in the electricity generation data.

```
RMSE 43753.71547566049
MAPE 2.8699996952065563
Accuracy 97.13000030479344
                           date   time         timestamp  test_predictions
2023-05-15 00:00:00  2023-05-15  00:00  2023-05-15 00:00      1.304279e+06
2023-05-15 01:00:00  2023-05-15  01:00  2023-05-15 01:00      1.268141e+06
2023-05-15 02:00:00  2023-05-15  02:00  2023-05-15 02:00      1.212346e+06
2023-05-15 03:00:00  2023-05-15  03:00  2023-05-15 03:00      1.119792e+06
2023-05-15 04:00:00  2023-05-15  04:00  2023-05-15 04:00      1.008560e+06
...                         ...    ...                ...               ...
2023-05-15 19:00:00  2023-05-15  19:00  2023-05-15 19:00      9.807618e+05
2023-05-15 20:00:00  2023-05-15  20:00  2023-05-15 20:00      1.010204e+06
2023-05-15 21:00:00  2023-05-15  21:00  2023-05-15 21:00      1.045572e+06
2023-05-15 22:00:00  2023-05-15  22:00  2023-05-15 22:00      1.044087e+06
2023-05-15 23:00:00  2023-05-15  23:00  2023-05-15 23:00      1.046514e+06

[24 rows x 4 columns]
```

Fig.7 ESM model performance

The best model parameters were used to fit the model on the entire dataset. This fitted model was then employed to make predictions for the next 24 hours after the available data. The forecasted values were obtained for each hour, allowing for a comprehensive understanding of electricity generation trends.
The ESM forecast model presented in this project provides valuable insights into electricity generation patterns. The accurate predictions for the next 24 hours allow for proactive decision-making, resource allocation, and load management in power systems. However, it is essential to note that the forecast model's performance may be influenced by factors such as data quality and the availability of historical data.
Overall, the developed ESM forecast model for electricity generation showcases the potential of time series analysis in optimizing power system operations and planning. It provides a foundation for future enhancements and applications in the field of electricity forecasting.
The provided code is integrated into a pipeline, where the resulting DataFrame (next_24_hours_df) is stored in a ClickHouse database.

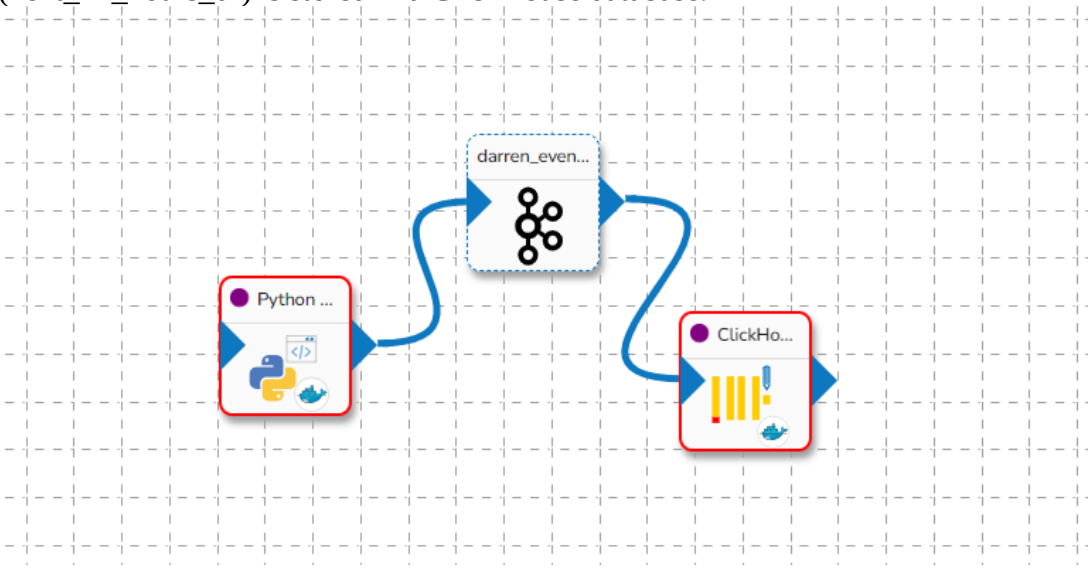


Fig 7.1 Forecasting pipeline

The provided code is integrated into a pipeline, where the resulting DataFrame (yesterday_data) is stored in a ClickHouse database.

To evaluate the performance and compare forecasted values with the actual values, the code snippet provided prepares the original data. Later, a SQL join query can be utilized to combine both tables for further analysis.

```
import datetime
import pandas as pd
import numpy as np
from clickhouse_driver import Client
def predict_time_series():
    client = Client(host='clickhouse.clickhouse', port='9000', database='db_darren_andrew', user='darren_andrew', password='Tg4246')

    # Execute query
    query = 'SELECT * FROM Electricity_production'
    results = client.execute(query)
    df = pd.DataFrame(results, columns=['date', 'respondent_name', 'respondent_name_id','time','type_name','type_name_id','value','value_units'])
```

```
# Close database connection
client.disconnect()
df['timestamp'] = pd.to_datetime(df['date'] + ' ' + df['time'])
df = df.set_index('timestamp')

# resample data to hourly frequency
df = df.resample('H').sum()

# handle missing values and outliers
df = df.replace(0, np.nan)
df = df.interpolate(method='linear', limit_direction='both')
df = df.fillna(method='bfill')
df['timestamp'] = df.index
df['date']=df['timestamp'].dt.strftime('%Y-%m-%d')
df['time']=df['timestamp'].dt.strftime('%H:%M')
df['timestamp']=df['timestamp'].dt.strftime('%Y-%m-%d %H:%M')
# Get yesterday's date
yesterday = datetime.datetime.now() - datetime.timedelta(days=2)
yesterday_start = yesterday.replace(hour=0, minute=0, second=0)
yesterday_end = yesterday.replace(hour=23, minute=59, second=59)

# Filter dataframe for yesterday's data
yesterday_data = df.loc[(df.index >= yesterday_start) & (df.index <= yesterday_end)]

return yesterday_data
```

The provided code is integrated into a pipeline, where the resulting DataFrame (yesterday_data) is stored in a ClickHouse database.

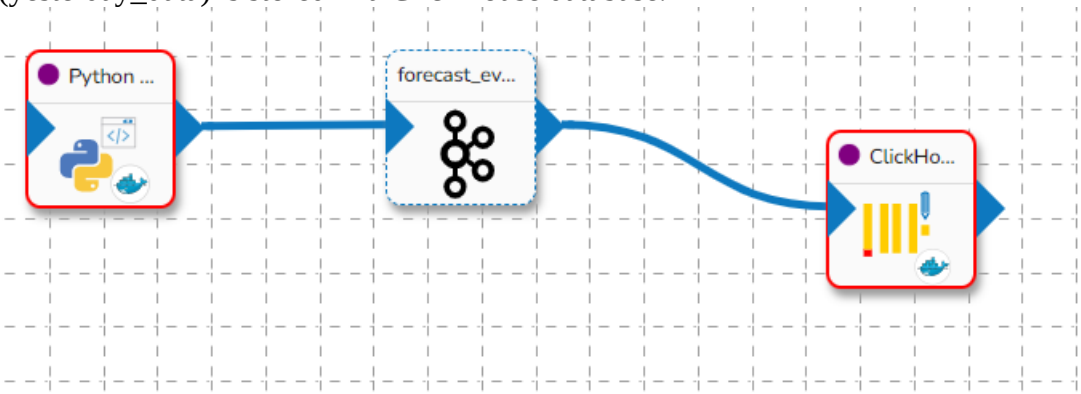


Fig.7.2 pipeline editor

8 Visualization

Let’s visualize by plotting charts and get more insights of the data. From here we’ll be using business story module of the BDB tool.

8.1 First collect the data using datasets from datacenter module.

Using the existing data connector click on the ‘+’ icon and then create a new data store.

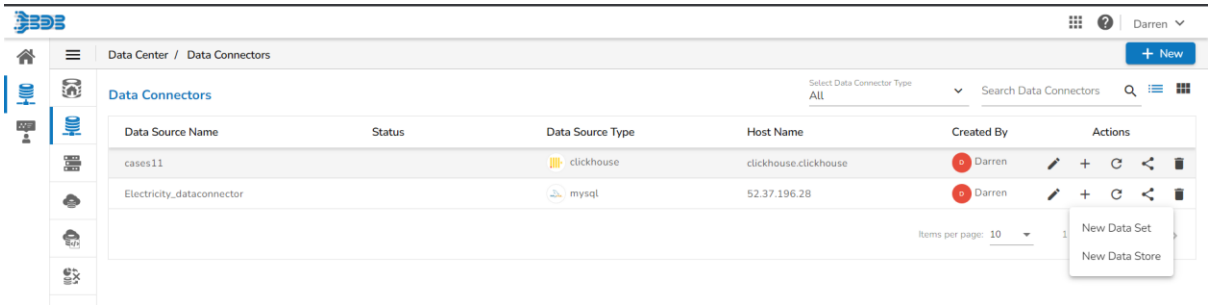


Fig 8.1 Creating a datastore

Then write an SQL query

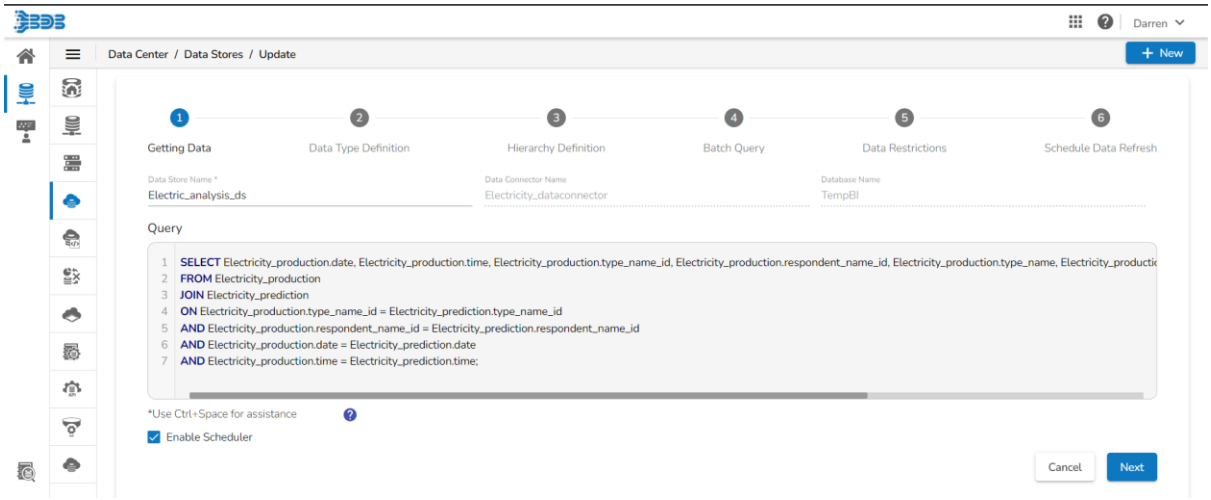


Fig 8.2 SQL query for datastore

Categorize the fields based on measures, dimensions and time.
Here I dragged date and time field from Dimensions to time.

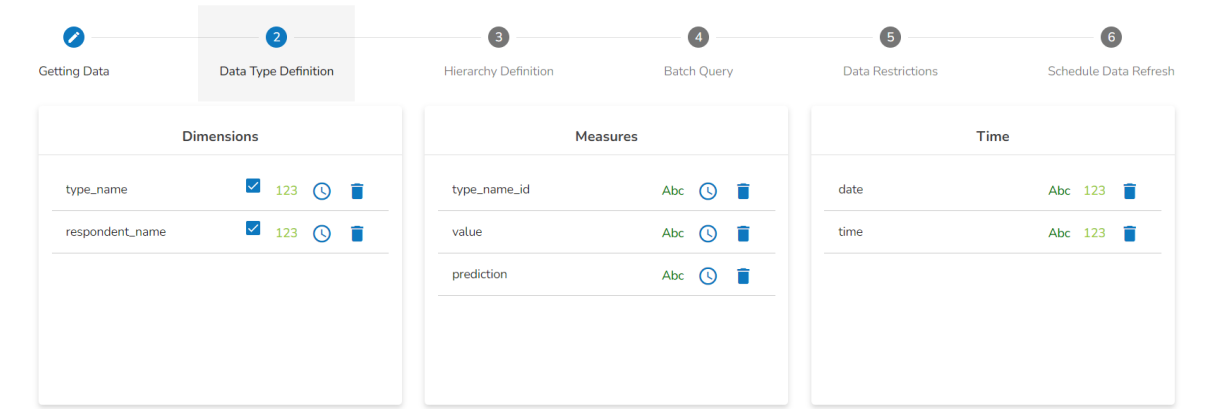


Fig.8.3 Datatype defination

On clicking next button, we can add hierarchy to the data,
By just date to the hierarchy, it automatically split date into year month and day. Moreover we can add another drill and add the respective fields respondent name and type name .

Since we do not require batch query and data retriction we can skip these steps.
At last we have Schedule data refresh

Since I want to refresh the data everyday since data is added to the data base everyday.
Set scheduler to daily and check tick on refresh now.

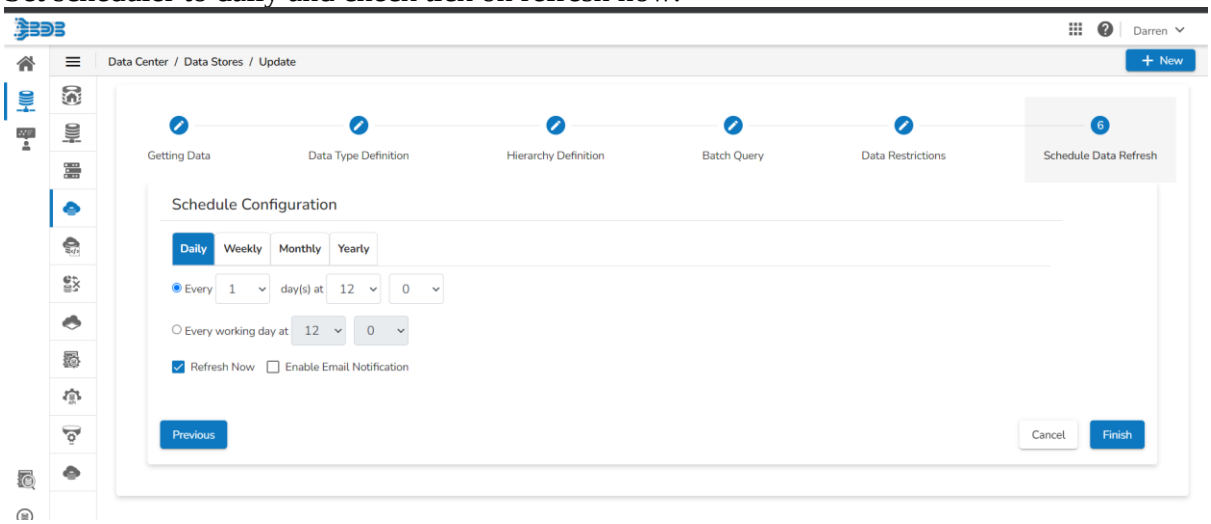


Fig 8.3 Schedule

Click on finish and now the data store will be created.

8.2 Business Story

Let's use this data store to create a business story.
In the home screen navigate to the business story.
A create story page will appear give a name and select the data store that we created using the above steps.

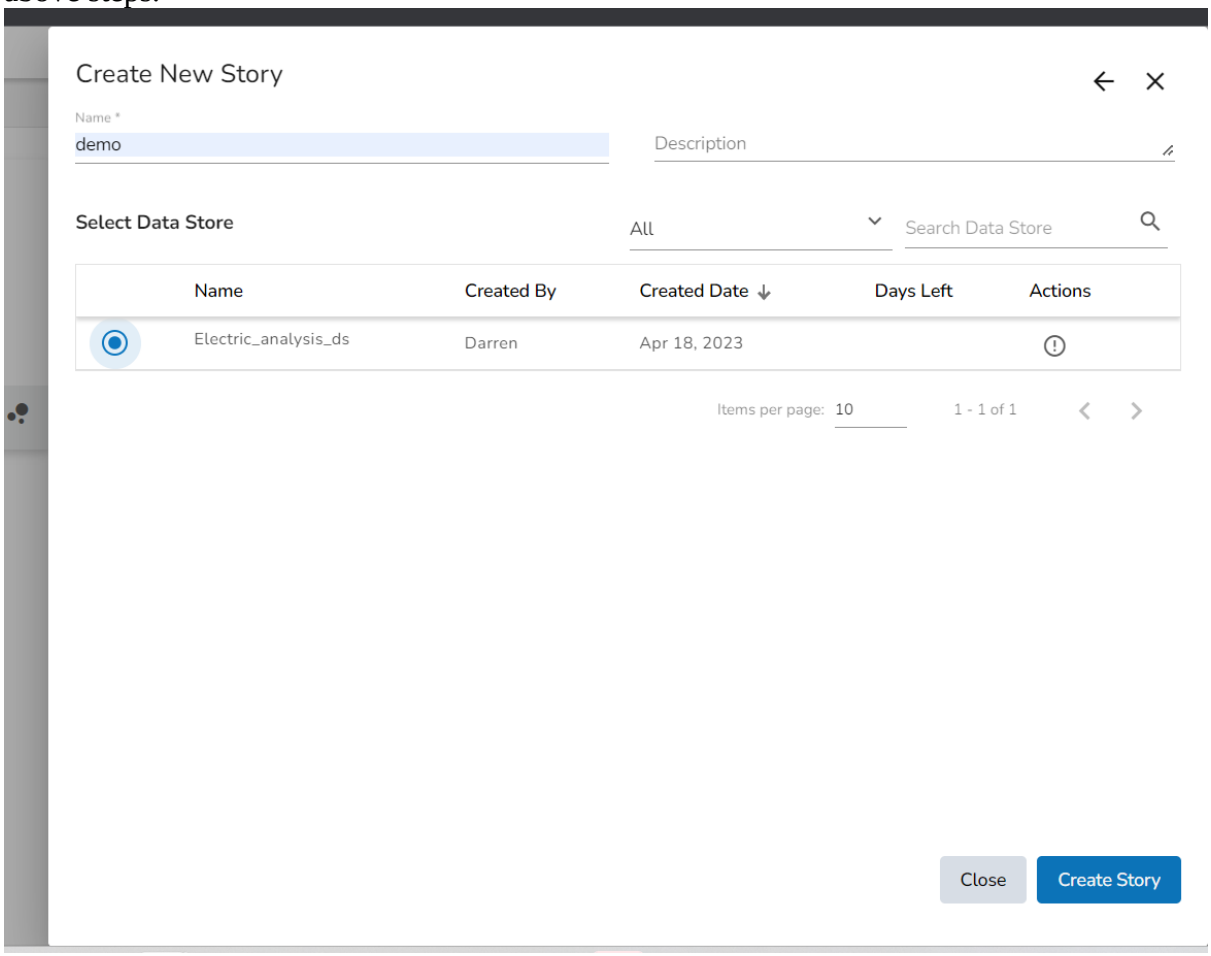


Fig 8.2.1 Creating a story

Click on create story.

The below image is the interface of the business story.

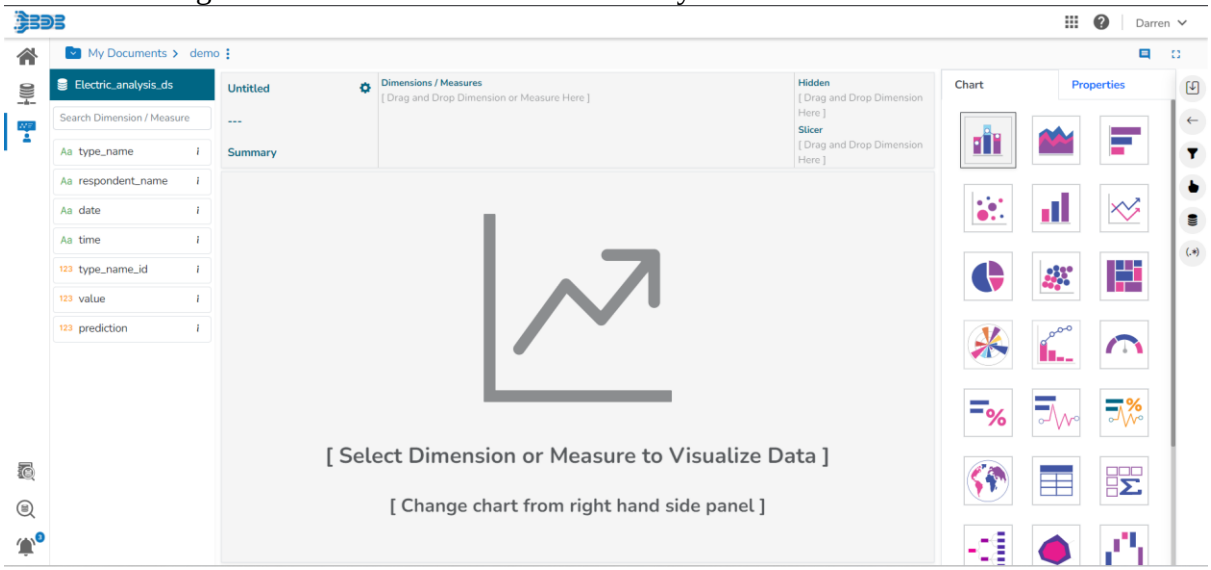


Fig 8.2.2 Story canvas

On the left plane we can see the fields of the data and on the right we can see the list of charts.

At first I'll plot a KPI.
On the right select the KPI chart then select the respondent_name field from the left pane.
In the below image we can see that a KPI has been plotted.

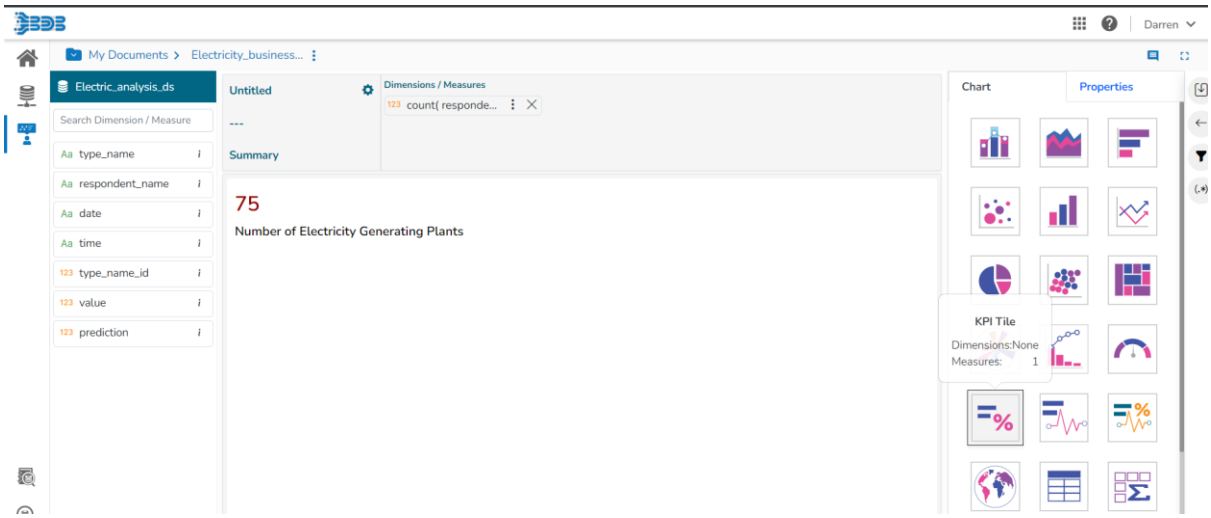
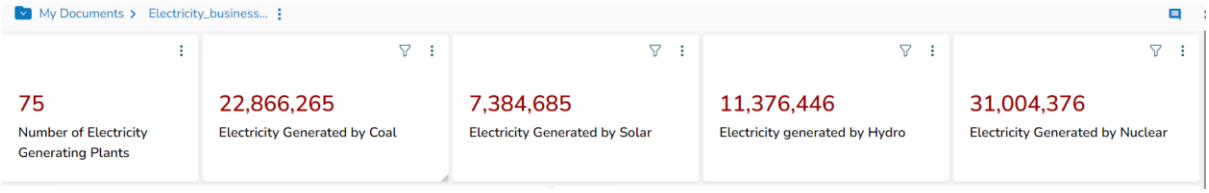


Fig.8.2.3 Ploting KPI's

Click on save.
Similarly lets plot other KPIs.



As we can see in total I have plotted 5 KPIs

Now let's plot a bar graph.
In this bar graph let us find the average electricity produced on each day.

Select date and value fields to the top pane.
Just beside the value there is a 3 dot icon, click on it and an window appears on the right side. Select the aggregation as mean.

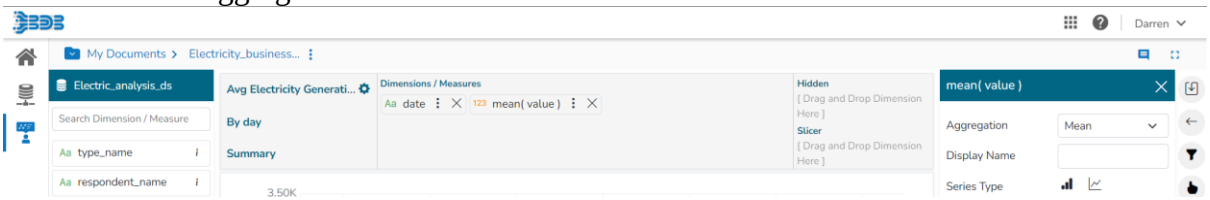


Fig.8.2.3 configuring graphs

Then select the bar graph from the charts menu on the right pane.

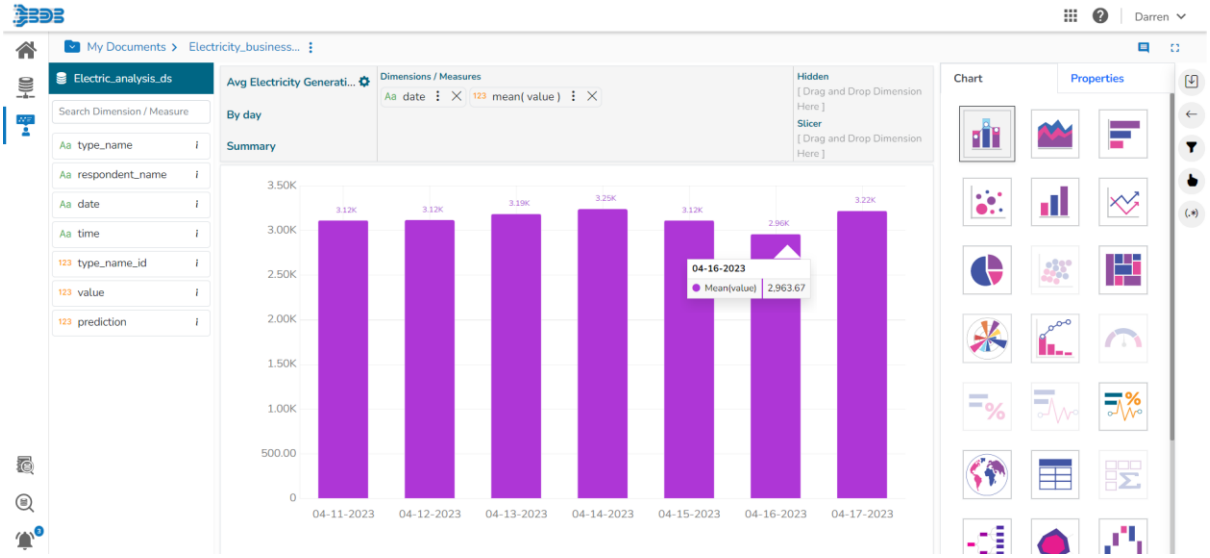


Fig.8.2.3 Bar Graphs

Similarly we can plot actual vs the predicted values from AutoML .
Select fields type_name, value and predicted as sum aggregation.

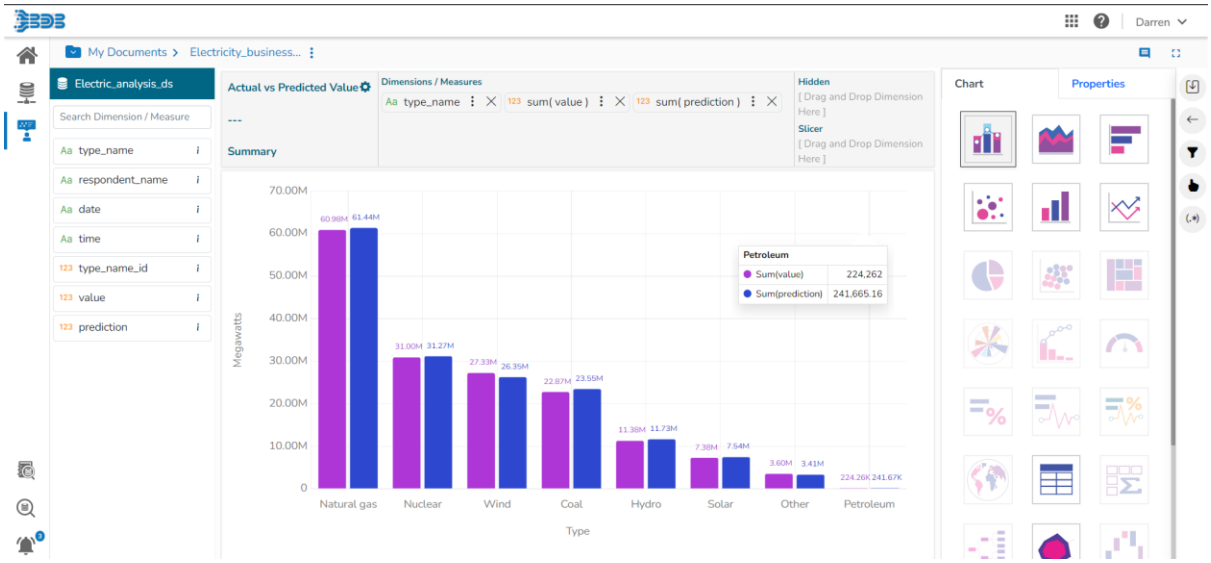


Fig.8.2.4 Actual VS Prediction

Now lets plot a horizontal bar graph.
Select fields respondent_name and value.
In the properties tab from the right pane. Configure the order, orderby and set limit to 10.
The following graph will show the top 10 Generation plants .

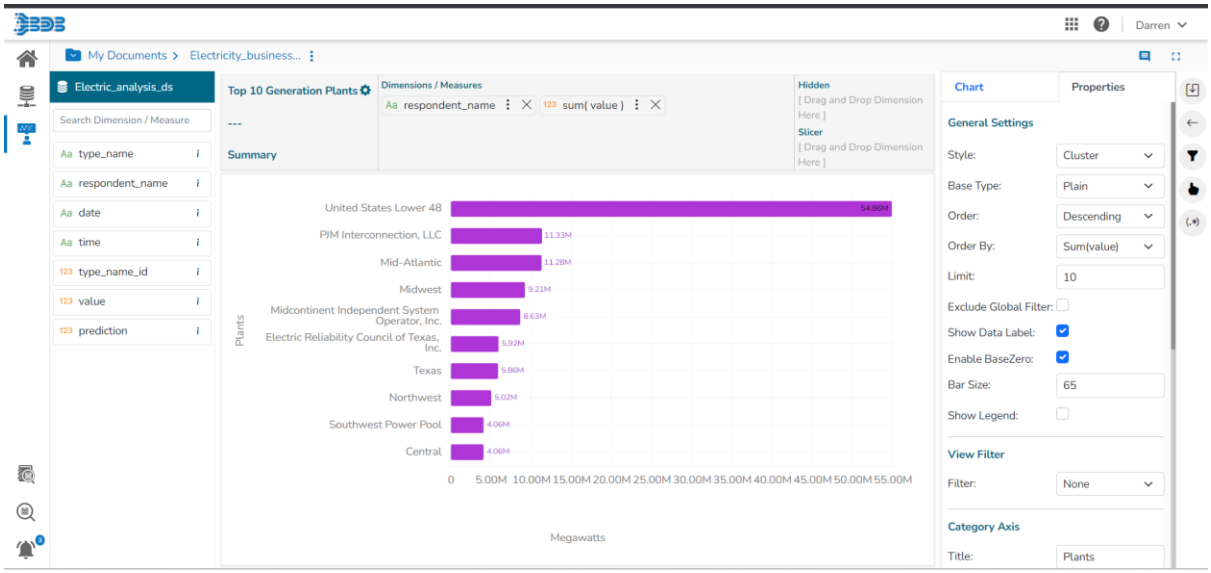


Fig.8.2.5 Respondent graph

Tree Map chart
Select fields type name and value.

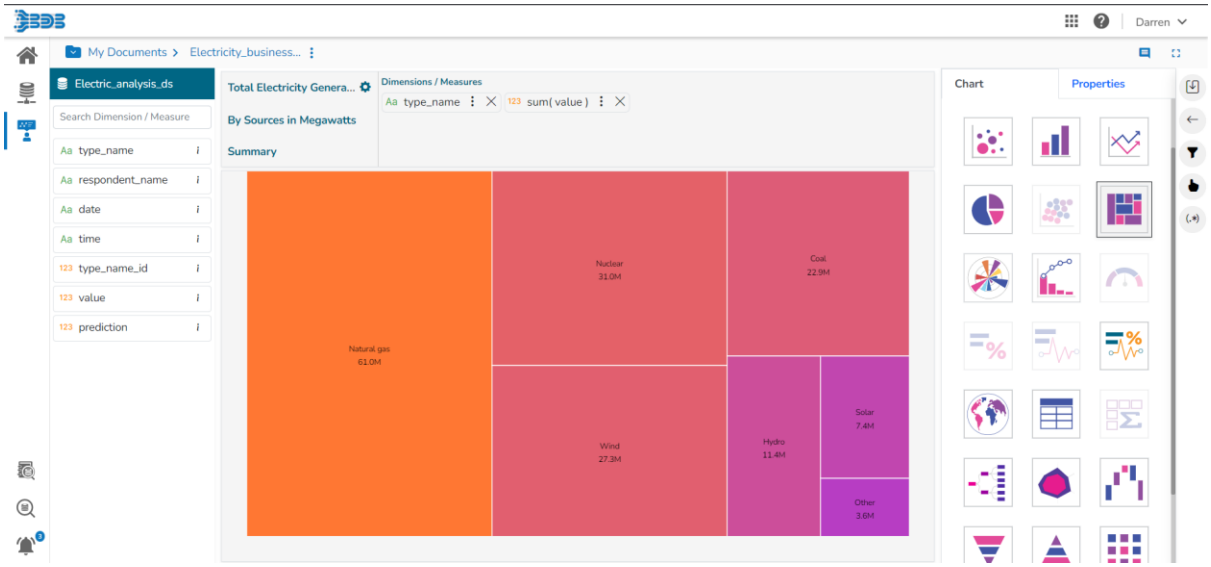


Fig.8.2.6 Tree map

Decomposition tree:
Select fields respondent_name , type_name and value.

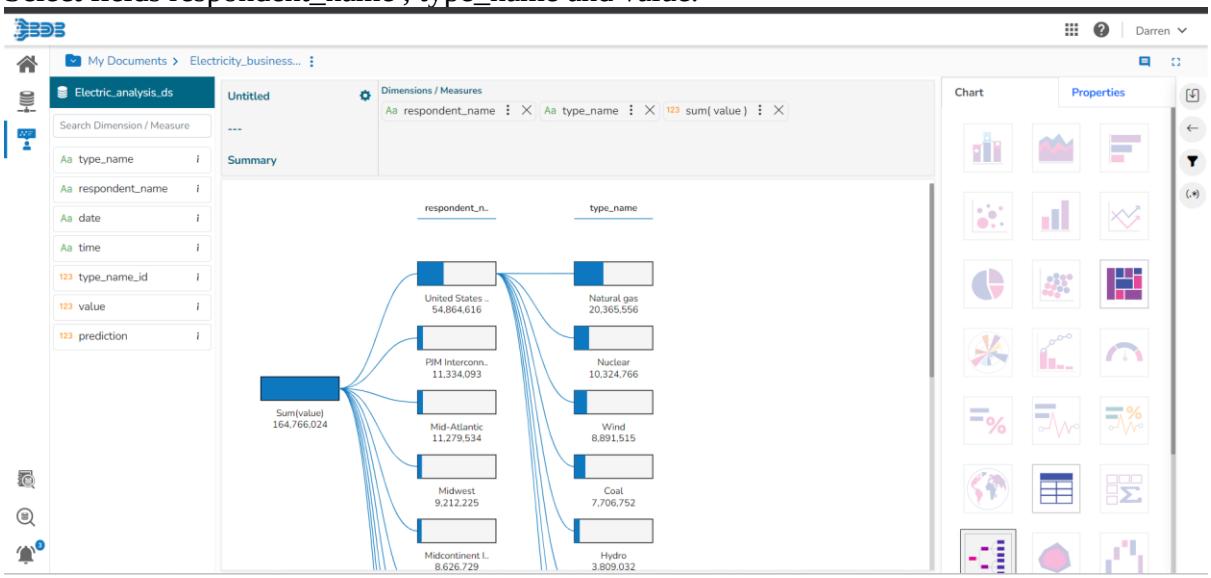


Fig.8.2.7 Decomposition

8.3 Business story for Prediction Data

Similarly lets create another business story to analyse Predictions done using Regression.
First lets create a Datastore using the below sql query.

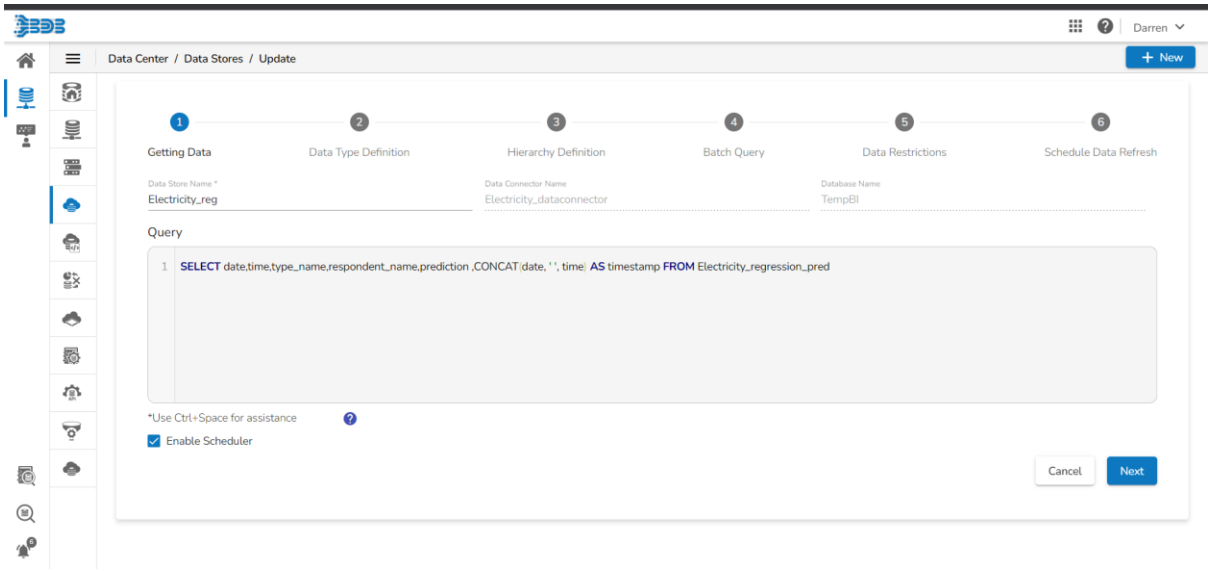


Fig.8.3.1 Datastore for regression data

Now create a story and select the above datastore.

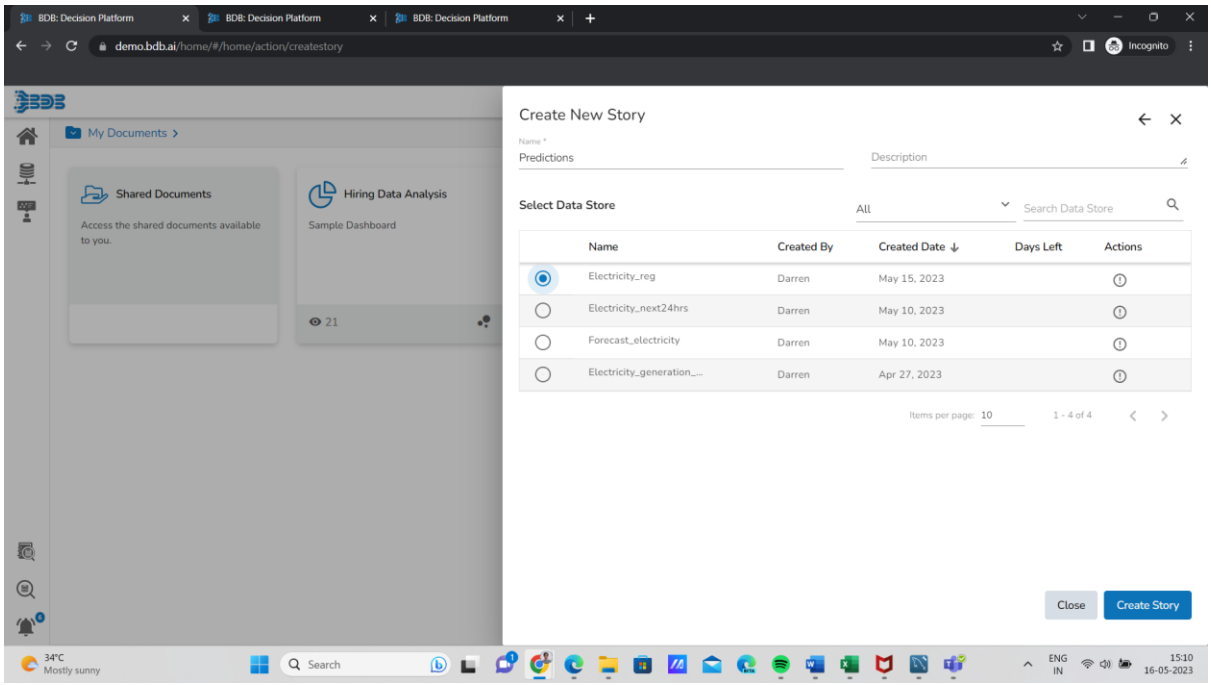


Fig.8.3.2 Creating story

Just like previous story let's plot few KPIs a column chart(Source vs Power generated), bar chart (Respondent vs Power generated) and a line chart(timestamp vs power generated)

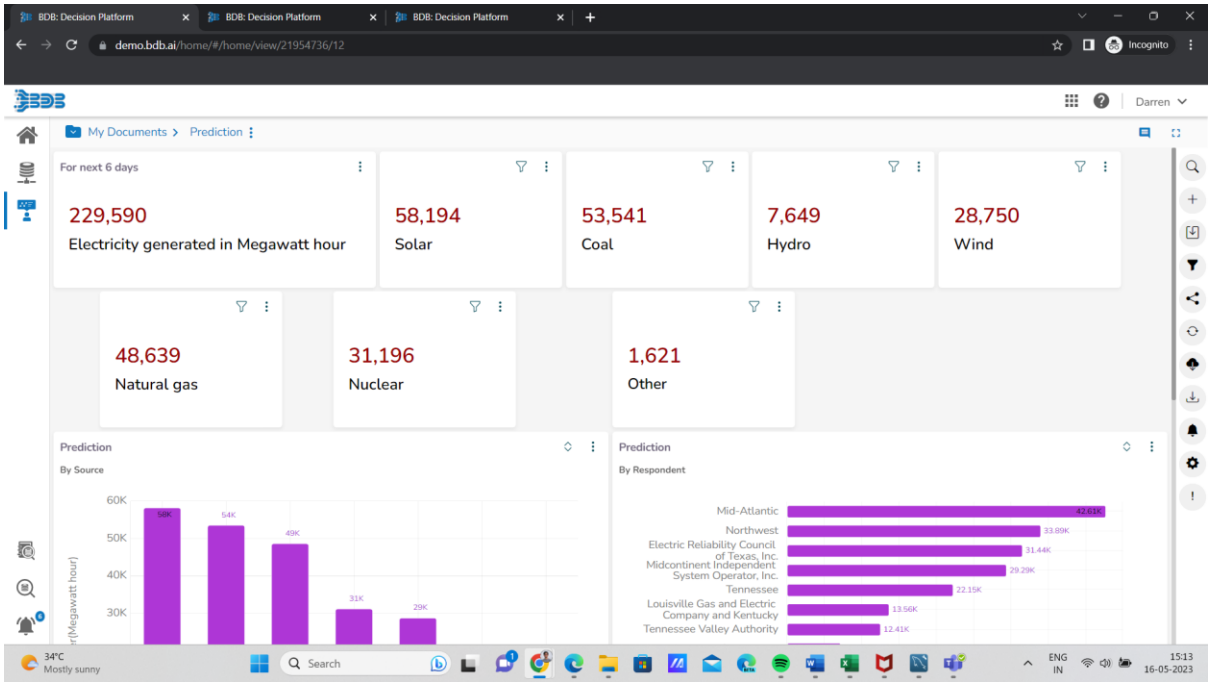


Fig.8.3.3 Story for Predicted values

The story displays information about the power that will be generated over the next 6 days. The model used to make these predictions is a random forest. A random forest is an ensemble learning method that combines multiple decision trees to make predictions. Decision trees are a type of machine learning algorithm that can be used to classify or predict values.

In the context of power generation, a random forest model can be used to predict the amount of power that will be generated over a given period of time. The model does this by training on historical data about power generation. The historical data includes information such as the time of day, the weather, and the amount of demand for power. The model uses this information to create a set of decision trees. Each decision tree makes a prediction about the amount of power that will be generated.

The random forest model then combines the predictions from the individual decision trees to make a final prediction about the amount of power that will be generated. The random forest model is a powerful tool that can be used to predict power generation. The model is accurate and reliable, and it can be used to make predictions over a long period of time.

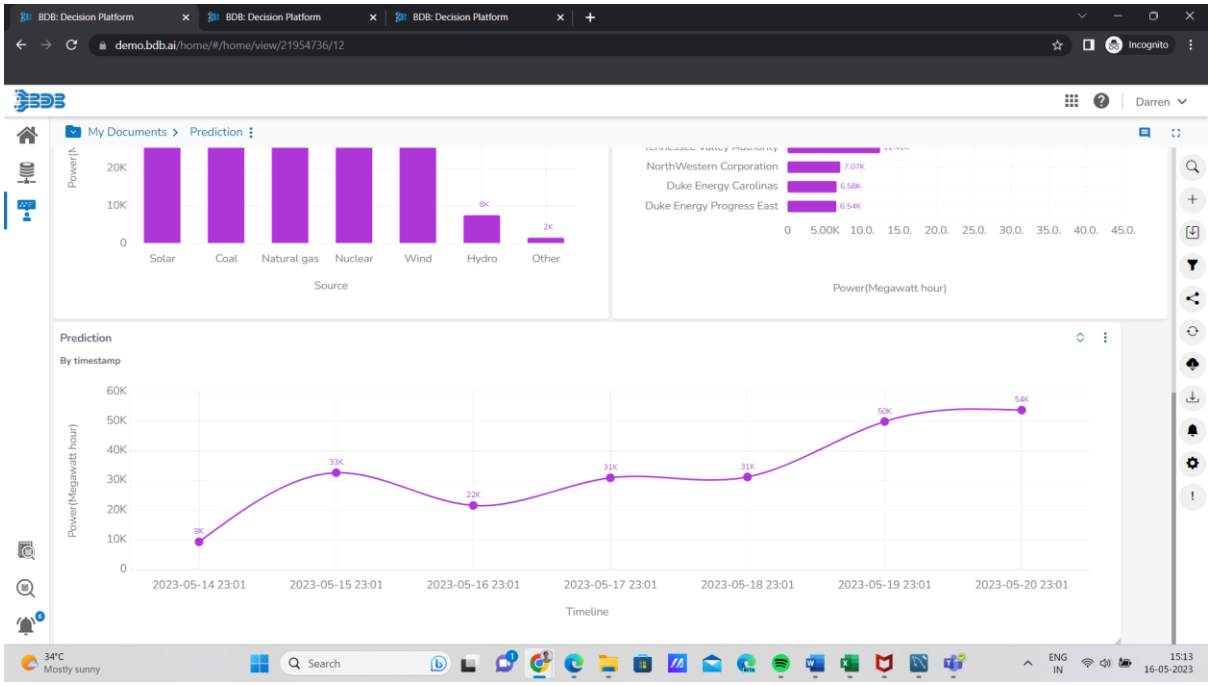


Fig.8.3.4 Timeline of Predicted values

8.4 Visualization for forecasting data (ESM)

The below two images creates a datastore that involves timeseries forecasting.

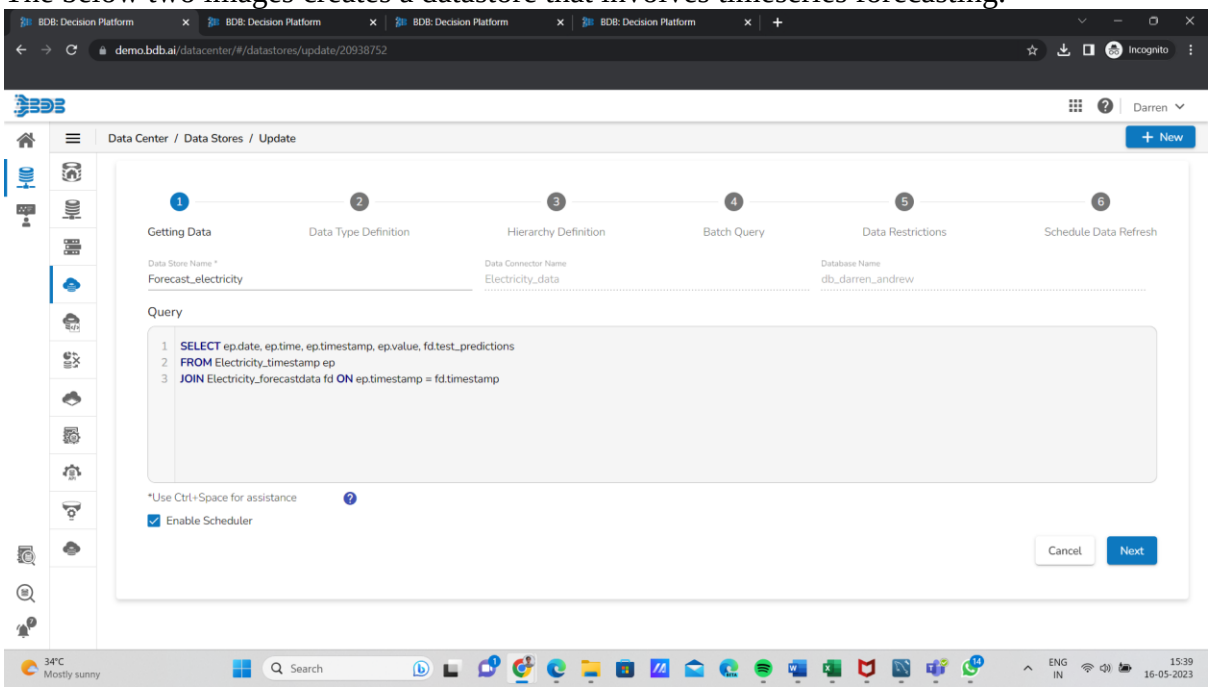


Fig. 8.4.1 Configuring Datastore for Forecasting data (Actual VS Forecast value)

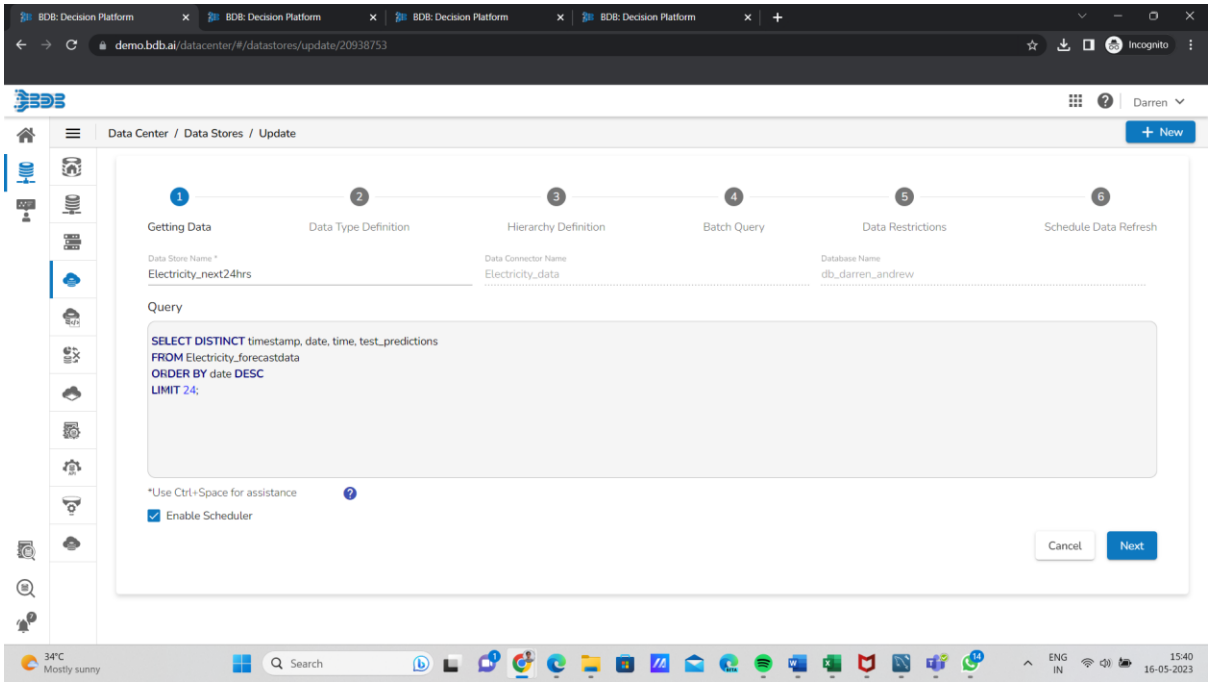


Fig. 8.4.2 Configuring Datastore for Forecasting data

To plot charts in the Dashboard Designer, the following steps are followed:

- Navigate to the Dashboard Designer interface.
- Click on the "New Dashboard" option to create a new dashboard.
- On the right pane, select "Data Connectors" from the available options.
- Click on the '+' icon near the data store to add a new data store.

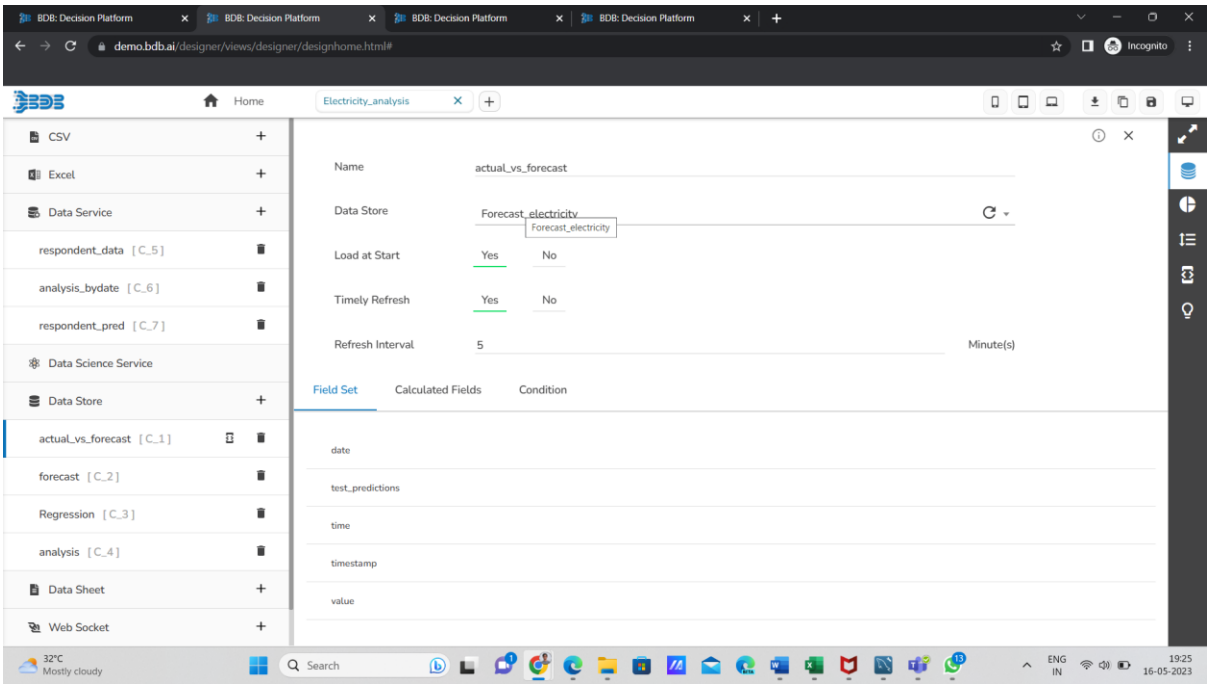


Fig.8.4.3 Configuring connections

- Choose the appropriate data store from the options provided.
- In the canvas area, drag and drop a timeseries chart component.
- Select the desired dataset from the available options.
- Choose the relevant fields from the dataset to be displayed on the chart.

- Click on the "Preview" button to visualize and preview the chart.

From the above steps we could plot a forecasting graph.
The below image depicts the dashboard.

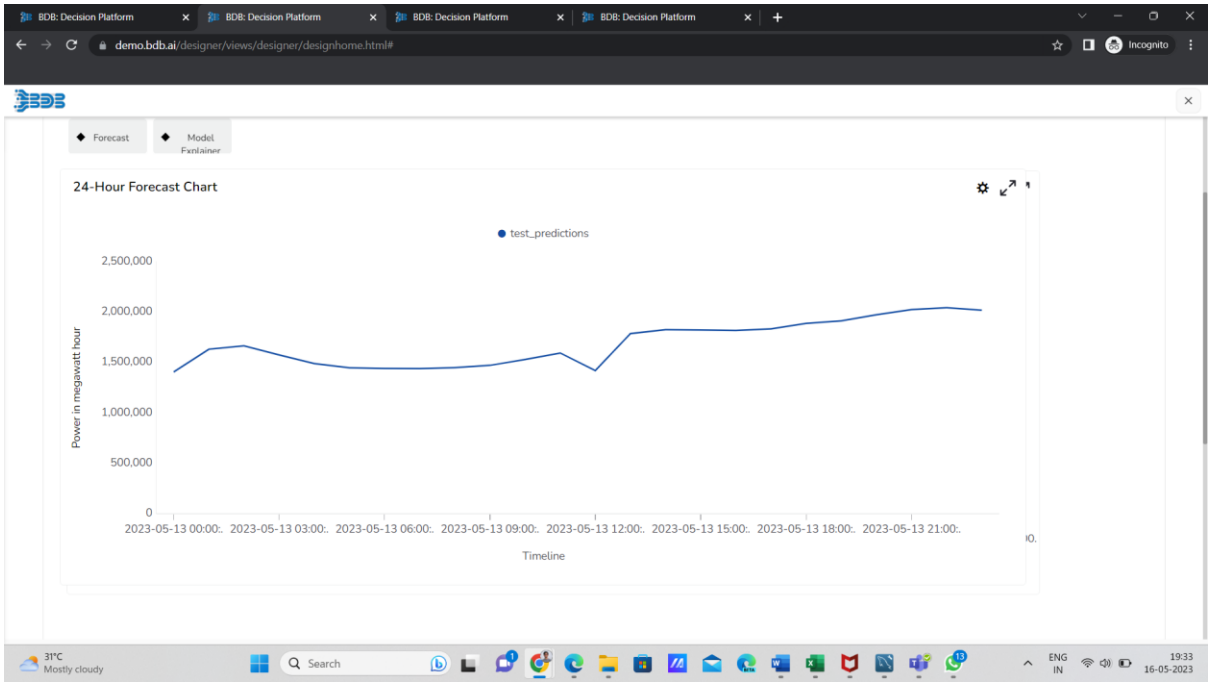


Fig.8.4.3 Forecast graph

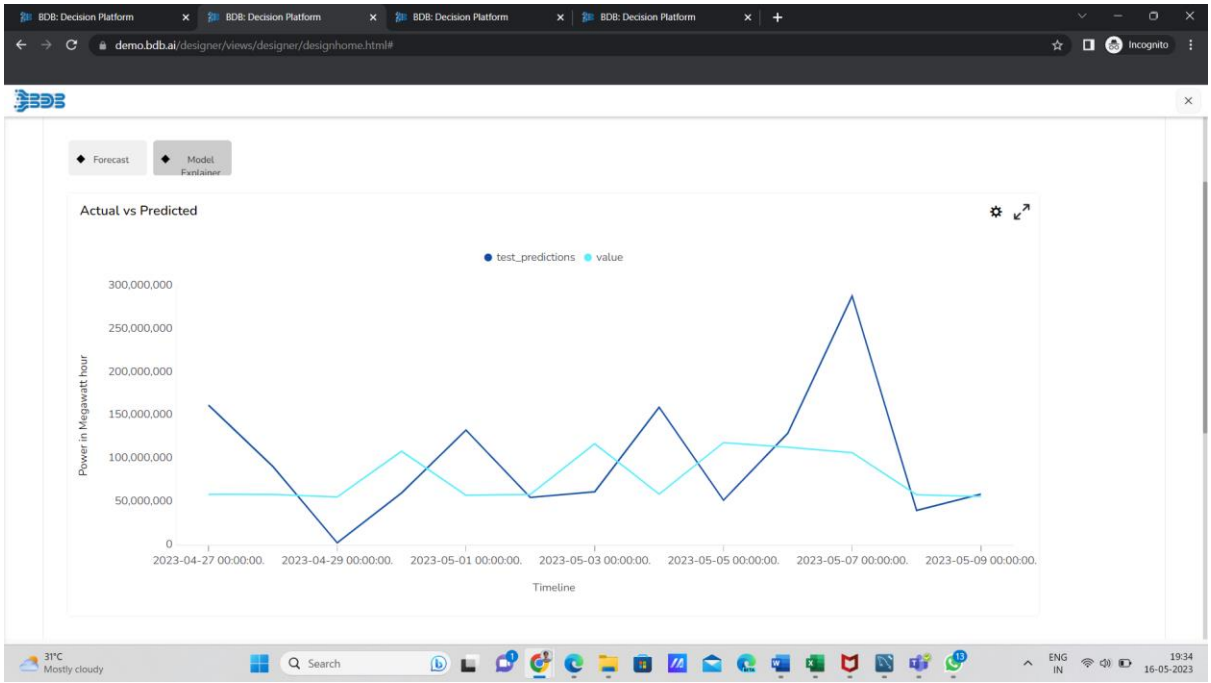


Fig.8.4.4 Actual VS Forecasted Graph

Similarly, in this dashboard, we can also plot the overview of the data and perform regression analysis. The steps mentioned earlier for plotting charts in the Dashboard Designer can be applied for these purposes as well. By selecting the appropriate chart types and datasets, we can create visualizations for data overview and regression analysis in the dashboard. The

flexibility of the Dashboard Designer allows us to customize and arrange the charts according to our specific requirements.

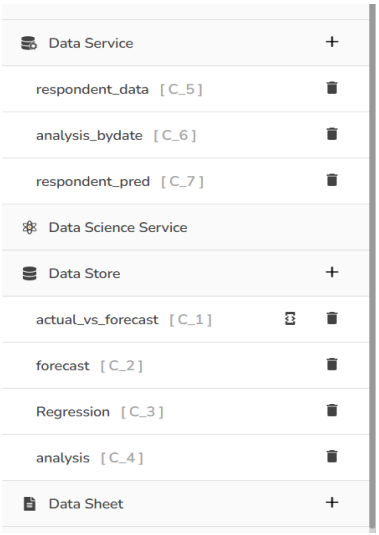


Fig.8.4.5 List of DataConnectors

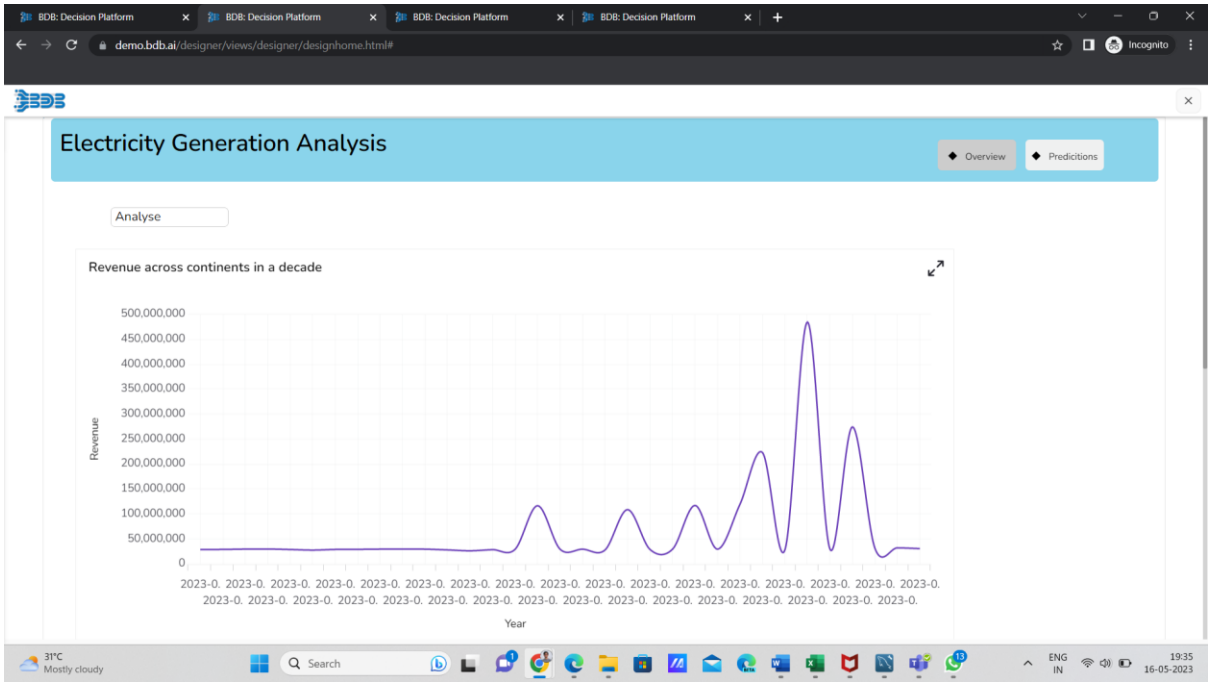


Fig.8.4.6 Overview of the data

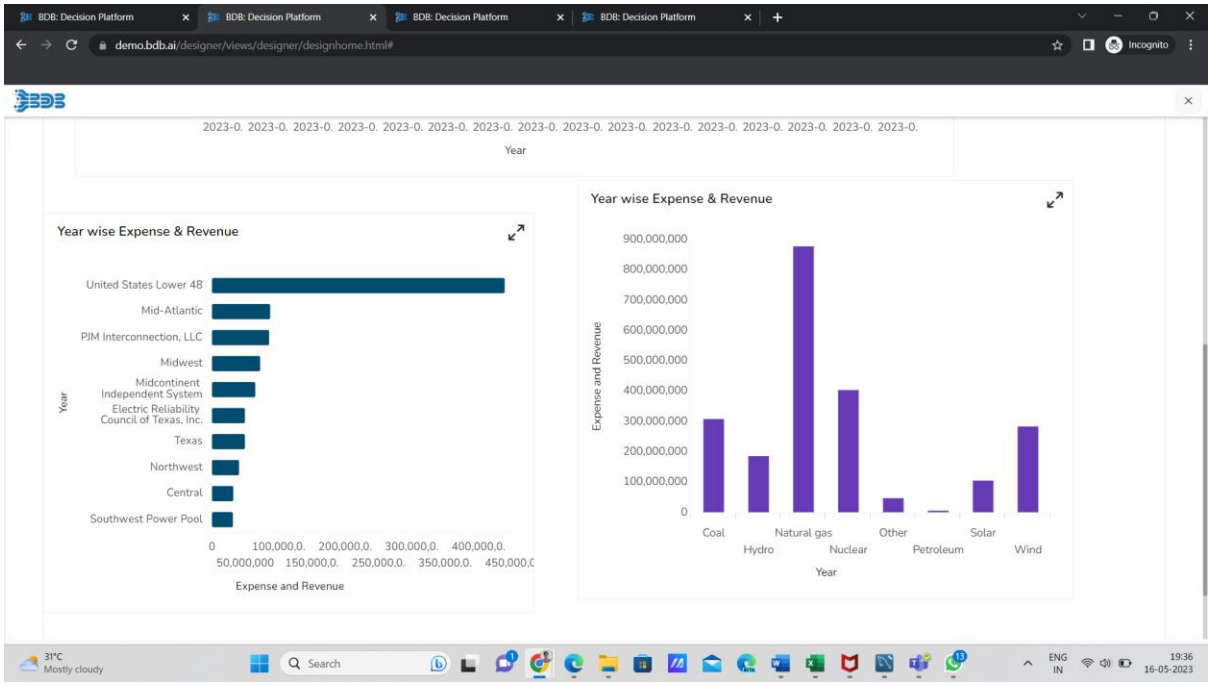


Fig.8.4.7 Overview of the data

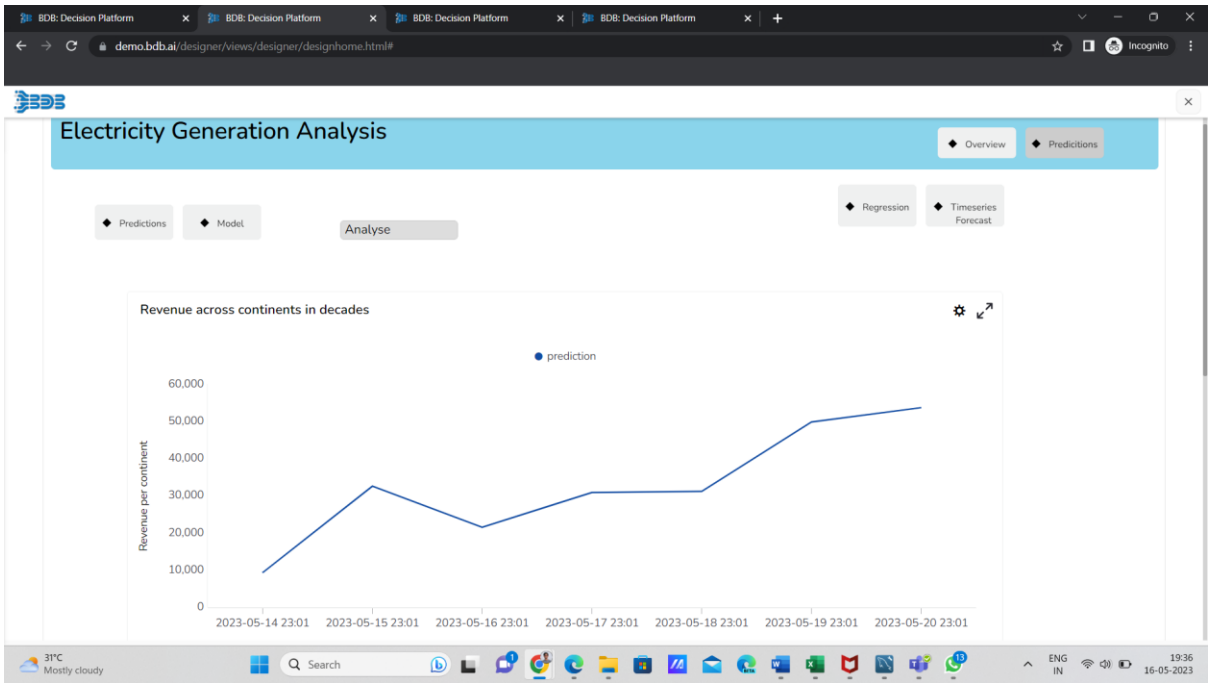


Fig.8.4.8 Regression analysis

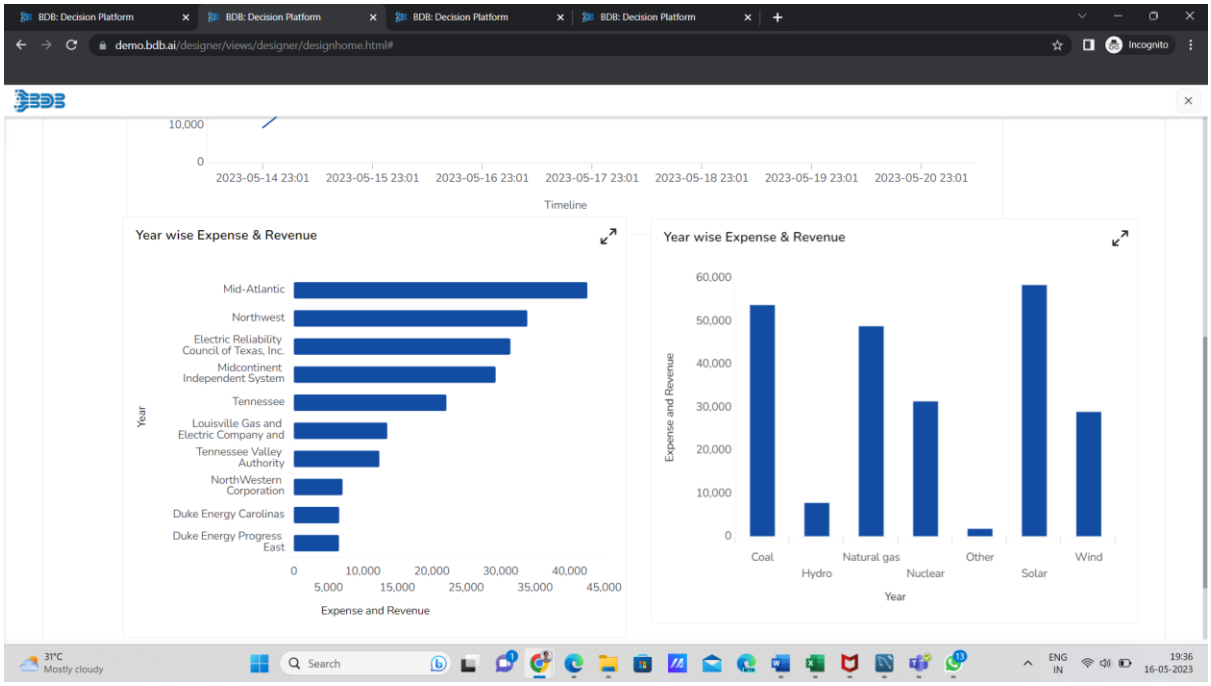


Fig.8.4.9 Regression analysis

Conclusion

In conclusion, the project aimed to develop a forecasting model for electricity generation using data extraction, data preparation, model training, and forecasting techniques. The project followed a pipeline-based approach to streamline the data processing and analysis tasks. Here are the key takeaways:

- Data Extraction and Preparation: The project started by extracting electricity generation data from the EIA website using their API. A Python script was utilized to scrape the data, clean it, and store it in a database for further analysis. The data was then prepared by encoding categorical variables and structuring it in a suitable format.

- Model Training: The prepared dataset was used to train a regression model using AutoML in DS Labs. The model selection was automated, and the best-performing models were recommended based on their RMSE scores. The selected model was registered and integrated into the pipeline for further use.

- Dataset Preparation for Forecasting: To generate predictions for the next six days, a dataset was created by replicating the existing data for multiple dates. The dataset was then fed into the AutoML component, which generated predicted values using the trained model. The predicted values were stored in a database for further analysis and visualization.

- Forecasting: The final stage of the project focused on forecasting electricity generation for the next 24 hours using Exponential Smoothing (ESM) as the forecasting technique. The historical data from the database was resampled to an hourly frequency, and missing values were handled through interpolation and backward filling. The ESM model was trained using the resampled data, and its performance was evaluated using RMSE as the evaluation metric.

The project demonstrated the end-to-end process of extracting, preparing, training, and forecasting electricity generation data. By utilizing pipeline-based tools and techniques, it streamlined the workflow and automated certain steps, such as model selection and data replication. The project's outcomes can be valuable for load management, resource planning, and decision-making processes in the power industry.

Overall, the project showcased the potential of data-driven approaches in the field of electricity generation forecasting and provided insights into the accuracy and reliability of the developed models. Further improvements and refinements can be made to enhance the forecasting accuracy and explore additional forecasting techniques for better predictions in the future.

Glossary:

Forecasting: The process of making predictions or estimates about future values based on historical data and patterns.
Data Extraction: The process of retrieving data from various sources such as websites, databases, APIs, etc.
Data Preparation: The process of cleaning, transforming, and structuring raw data into a suitable format for analysis and modeling.
Model Training: The process of building a predictive model by feeding historical data to an algorithm to learn patterns and relationships.
Regression Model: A type of statistical model used to predict a continuous numerical value, such as electricity generation, based on input variables.
AutoML: Automated Machine Learning, a set of tools and techniques that automate the process of selecting, training, and evaluating machine learning models.
RMSE (Root Mean Square Error): A commonly used metric to measure the accuracy of a predictive model by calculating the square root of the average squared differences between predicted and actual values.
Dataset: A collection of structured data used for analysis and modeling.
Interpolation: A technique used to estimate missing values in a dataset by inferring values based on known data points.
Exponential Smoothing: A time series forecasting method that assigns exponentially decreasing weights to past observations to predict future values.
Workflow: The sequence of tasks or steps involved in completing a specific project or process.
Evaluation Metric: A measure used to assess the performance or accuracy of a predictive model, such as RMSE, MAE (Mean Absolute Error), or R-squared.
Fossil Fuels: Non-renewable energy sources, such as coal, oil, and natural gas, formed from the remains of ancient plants and animals.
Renewable Energy Sources: Energy sources derived from natural resources that are replenished over a short period, such as solar power, wind power, hydroelectricity, etc.

References:

https://www.bdb.ai/
https://towardsdatascience.com/
eia.gov/opendata/browser/electricity/rto/fuel-type-data
www.geeksforgeeks.org