# MACHINE LEARNING MODEL DEPLOYMENT WITH IBM CLOUD WATSON STUDIO

## FINAL PROJECT SUBMISSION

**Mentor:**

**Nishanthini**

**(nishanthi.ac.in)**

**TEAM MEMBERS:**

**1.Anseera J (au7145714521104004)**

**2.Abinaya K (aut21bcs001)**

**3.Sangavi B (au714521104043)**

**4.Soumeshver M (au714521104049)**

**5.Mageshwaran M (aut21bcs005)**

**6.RanjithKumar M.M (aut21bcs006)**

## Problem Statement:

Become a wizard of predictive with IBM Cloud Watson Studio. Train machine learning models to predict outcomes in real-time. Deploye the models as web services and integrate them into your applications. Unlock the magic of data-driven insights and make informed decisions like never before.

## Abstract:

Explore the power of machine learning with IBM Cloud Watson Studio. This project delves into the end-to-end process of model development, from data preparation to deployment, showcasing the efficiency and collaborative features of IBM Cloud Watson Studio. Discover how to harness the platform's tools and capabilities to create, train, evaluate, and deploy machine learning models, enabling data-driven decision-making for organizations of all sizes.

## Introduction:

In the era of data-driven decision-making, the integration of machine learning models has become a crucial asset for organizations seeking to derive valuable insights and make informed business decisions. IBM Cloud Watson Studio, a comprehensive and powerful platform, facilitates the development and deployment of sophisticated machine learning models. With its robust suite of tools and services, Watson Studio offers a seamless and collaborative environment for data scientists, developers, and domain experts to build, train, and deploy machine learning models at scale. Leveraging Watson Studio's intuitive interface and advanced features, businesses can harness the potential of their data to drive innovation, enhance customer experiences, and achieve significant operational efficiencies. This introduction serves as a gateway to exploring the rich capabilities of IBM Cloud Watson Studio, unlocking a world of possibilities in the realm of machine learning model development and deployment.

In the rapidly evolving landscape of artificial intelligence and data analytics, IBM Cloud Watson Studio stands as a cutting-edge platform for developing and deploying robust machine learning models. Offering an array of integrated tools and services, Watson Studio facilitates end-to-end model development, from data exploration and preprocessing to model training and deployment. Its collaborative and scalable environment empowers data professionals to expedite the development process and derive valuable insights from complex datasets. With a focus on simplifying complex tasks, Watson Studio is a game-changer for organizations seeking to leverage the power of machine learning to drive innovation and achieve tangible business results.

## Project Objective:

The objective of this project is to deploy a machine learning model that predicts customer churn in a subscription-based service. Through the analysis of historical customer data, the aim is to identify patterns and factors that contribute to customer churn, enabling the business to take proactive measures for customer retention. The primary objective is to successfully deploy a trained machine learning model on the IBM Cloud platform, making it accessible through APIs or web services. This ensures that the model can be used for making predictions on new data.

## Documentation

Outline the project's objective, design thinking process, and development phases.

Describe the predictive use case, dataset selection, model training, deployment process, and integration steps.

Explain how the deployed model can be accessed and utilized for real- time predictions.

## Submission

Share the GitHub repository link containing the project's code and files

Provide instructions on how to deploy and use the machine learning model as a web service.

Include example API requests for making prediction.

# INNOVATION AND DESIGN THINKING

## Introduction:

Machine learning is revolutionizing the way we extract insights and make predictions from data, and IBM Cloud Watson Studio is at the forefront of this transformation. Watson Studio is a powerful platform that empowers data scientists and developers to create, train, and deploy machine learning models with ease.

With Watson Studio, we can leverage the capabilities of IBM Cloud, including robust data storage and management, to access and preprocess our data seamlessly. The platform offers a collaborative environment where teams can work together on data projects, making it an ideal choice for organizations looking to harness the potential of artificial intelligence and machine learning.

Whether we are a seasoned data scientist or just beginning your journey into the world of machine learning, IBM Cloud Watson Studio provides the tools and resources you need. From data exploration and feature engineering to model development and deployment, it streamlines the entire machine learning lifecycle. Additionally, Watson Studio's integration with IBM Watson AI services enables you to infuse AI capabilities into your applications, unlocking new possibilities for innovation.

In this guide, we'll explore the key features and benefits of IBM Cloud Watson Studio, walk through the steps to develop and deploy machine learning models, and demonstrate how this platform can accelerate your data-driven projects. So, let's embark on a journey to harness the power of AI and machine learning with IBM Cloud Watson Studio.

## Problem Statement Revisited:

Become a wizard of predictive with IBM Cloud Watson Studio. Train machine learning models to predict outcomes in real-time. Deploye the models as web services and integrate them into your applications. Unlock the magic of data- driven insights and make informed decisions like never before**.**

## Design Thinking Refinement :

### 1. Identify the Problem:

Understand the specific challenges and pain points in deploying machine learning models with Watson Studio. This could include issues related to scalability, version control, monitoring, or automation.

## 2. Understanding the Problem:

Start by clearly defining the problem it want to solve with machine learning. Understand the business context and goals to design an effective solution.

## 3. Define Goals:

Define what you want to achieve with the innovation. For example, you might aim to streamline the deployment process, improve model monitoring, or enhance collaboration among data scientists and DevOps teams.

## 4. Data Collection and Preparation:

Gather relevant data for the problem and preprocess it to ensure it's suitable for training models. Watson Studio provides tools for data cleansing, transformation, and integration.

## 5. Assess Current Tools and Technologies:

Evaluate the existing tools and technologies within Watson Studio for model deployment. Identify their strengths and weaknesses.

## 6. Exploratory Data Analysis (EDA):

Use visualizations and statistical analysis to understand patterns and relationships within the data. EDA helps in making informed decisions about feature selection and engineering.

## 7. Feature Engineering:

Engineer meaningful features from the data that can help improve model performance. Watson Studio assists in creating and selecting features for training the models.

## 8. Leverage Automation:

Implement automation to simplify the deployment process. Use tools like Kubernetes or Docker for containerization, and CI/CD pipelines for continuous integration and deployment.

## 9. Selection and Training:

Choose appropriate machine learning algorithms based on the problem and data. Train and evaluate various models to determine the most suitable one for your task.

## 10. Hyperparameter Tuning:

Optimize the performance of the chosen model by tuning hyperparameters using techniques like grid search or random search. Watson Studio supports hyperparameter tuning to enhance model accuracy.

## 11. Version Control:

Implement robust version control mechanisms for both code and models. This ensures traceability and reproducibility.

## 12. Model Evaluation and Validation:

Assess the model's performance using validation techniques like cross-validation, and evaluate metrics such as accuracy, precision, recall, etc. Adjust the model as needed.

## 13. Deployment and Integration:

Once the model is trained and validated, deploy it using Watson Studio. Integrate the model into your application or workflow for real-time predictions.

## 14. Collaboration:

Foster collaboration between data scientists and DevOps teams by creating an integrated environment within Watson Studio. Ensure seamless communication and knowledge sharing.

## 15. Security and Compliance:

Security measures to protect sensitive data and ensure compliance with regulatory requirements.

## 16. Scalability:

For scalability to handle large-scale deployments. Use cloud resources efficiently and consider auto-scaling options.

## 17. Feedback Loop:

Create a feedback loop for continuous improvement. Collect feedback from users and stakeholders to refine our deployment innovation.

## 18. Cost Optimization:

Optimize costs by monitoring resource usage and implementing cost-saving me as teams

## 19. Stay Updated:

Monitor the evolving landscape of machine learning deployment and adapt our innovation accordingly.

## 20. Monitoring and Maintenance:

Continuously monitor the deployed model's performance and retrain or update it as needed to ensure it remains effective and accurate over time

# DEVELOPMENT PART-1

## Introduction:

IBM Watson Studio is an integrated development environment (IDE) designed to streamline the process of building, training, and deploying AI models and applications. It is part of IBM's suite of AI tools and services and is aimed at enabling data scientists, developers, and domain experts to collaborate effectively and efficiently in creating and deploying machine learning and deep learning models.

Machine learning model deployment is a critical phase in the machine learning lifecycle, involving the integrationof trained models into real-world applications for making predictions or decisions based on new data. IBM CloudWatson Studio provides a powerful and user-friendly platform for deploying machine learning models effectivelyand efficiently. Leveraging the comprehensive capabilities of IBM Cloud Watson Studio for model deployment enables organizations to scale AI applications and deliver actionable insights.

## 1. Define the Predictive Use Case:

In this initial stage, you should have a clear understanding of the problem you aim to solve. For instance, if your use case is customer churn prediction, you need to specify what "churn" means in your context and the business implications.

## 2. Select a Relevant Dataset:

Carefully choose a dataset that aligns with your predictive use case. For customer churn prediction, your dataset might include information on customers, their interactions, demographics, purchase history, and whether they churned or not.

## 3. Access IBM Cloud Watson Studio:

Log in to your IBM Cloud account and access Watson Studio. This is your hub for all your machine learning activities.

## 4. Import the Dataset:

Use Watson Studio's data management tools to import your dataset. You can upload data files in various formats or connect to external data sources if your data is not already in Watson Studio.

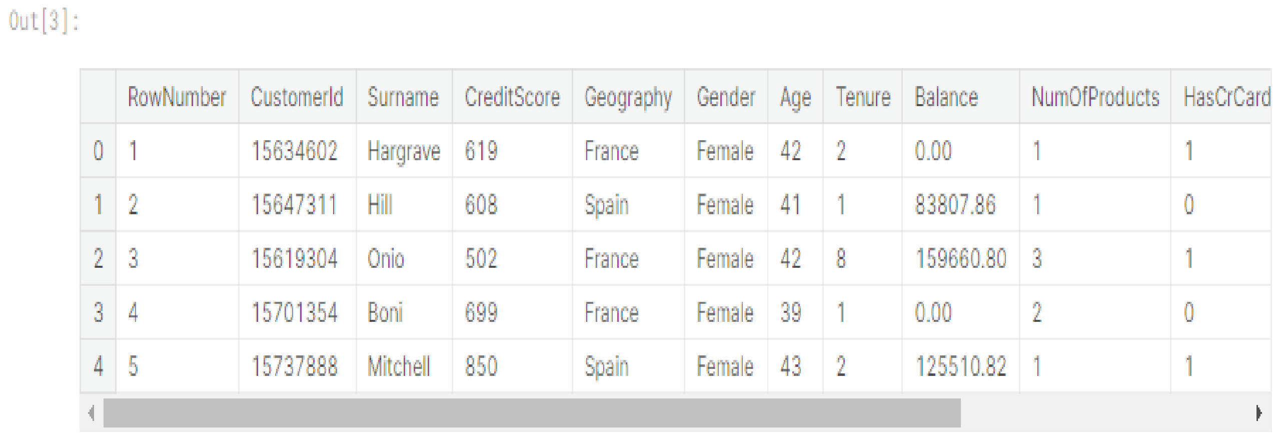
| R.N | Cus.Id | Surname | C.Score | Geograph | Gender | Age | Tenure | Balance | N.pro | Hashcrd | A.Mem | Estimated | Exited |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0 | 1 | 1 | 1 | 101348.9 | 1 |
| 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 | 112542.6 | 0 |
| 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.8 | 3 | 1 | 0 | 113931.6 | 1 |
| 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0 | 2 | 0 | 0 | 93826.63 | 0 |
| 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.8 | 1 | 1 | 1 | 79084.1 | 0 |
| 6 | 15574012 | Chu | 645 | Spain | Male | 44 | 8 | 113755.8 | 2 | 1 | 0 | 149756.7 | 1 |
| 7 | 15592531 | Bartlett | 822 | France | Male | 50 | 7 | 0 | 2 | 1 | 1 | 10062.8 | 0 |
| 8 | 15656148 | Obinna | 376 | Germany | Female | 29 | 4 | 115046.7 | 4 | 1 | 0 | 119346.9 | 1 |
| 9 | 15792365 | He | 501 | France | Male | 44 | 4 | 142051.1 | 2 | 0 | 1 | 74940.5 | 0 |
| 10 | 15592389 | H? | 684 | France | Male | 27 | 2 | 134603.9 | 1 | 1 | 1 | 71725.73 | 0 |
| 11 | 15767821 | Bearce | 528 | France | Male | 31 | 6 | 102016.7 | 2 | 0 | 0 | 80181.12 | 0 |
| 12 | 15737173 | Andrews | 497 | Spain | Male | 24 | 3 | 0 | 2 | 1 | 0 | 76390.01 | 0 |
| 13 | 15632264 | Kay | 476 | France | Female | 34 | 10 | 0 | 2 | 1 | 0 | 26260.98 | 0 |
| 14 | 15691483 | Chin | 549 | France | Female | 25 | 5 | 0 | 2 | 0 | 0 | 190857.8 | 0 |
| 15 | 15600882 | Scott | 635 | Spain | Female | 35 | 7 | 0 | 2 | 1 | 1 | 65951.65 | 0 |
| 16 | 15643966 | Goforth | 616 | Germany | Male | 45 | 3 | 143129.4 | 2 | 0 | 1 | 64327.26 | 0 |
| 17 | 15737452 | Romeo | 653 | Germany | Male | 58 | 1 | 132602.9 | 1 | 1 | 0 | 5097.67 | 1 |
| 18 | 15788218 | Henderso | 549 | Spain | Female | 24 | 9 | 0 | 2 | 1 | 1 | 14406.41 | 0 |
| 19 | 15661507 | Muldrow | 587 | Spain | Male | 45 | 6 | 0 | 1 | 0 | 0 | 158684.8 | 0 |
| 20 | 15568982 | Hao | 726 | France | Female | 24 | 6 | 0 | 2 | 1 | 1 | 54724.03 | 0 |
| 21 | 15577657 | McDonald | 732 | France | Male | 41 | 8 | 0 | 2 | 1 | 1 | 170886.2 | 0 |
| 22 | 15597945 | Dellucci | 636 | Spain | Female | 32 | 8 | 0 | 2 | 1 | 0 | 138555.5 | 0 |
| 23 | 15699309 | Gerasimov | 510 | Spain | Female | 38 | 4 | 0 | 1 | 1 | 0 | 118913.5 | 1 |
| 24 | 15725737 | Mosman | 669 | France | Male | 46 | 3 | 0 | 2 | 0 | 1 | 8487.75 | 0 |
| 25 | 15625047 | Yen | 846 | France | Female | 38 | 5 | 0 | 1 | 1 | 1 | 187616.2 | 0 |
| 26 | 15738191 | Maclean | 577 | France | Male | 25 | 3 | 0 | 2 | 0 | 1 | 124508.3 | 0 |
| 27 | 15736816 | Young | 756 | Germany | Male | 36 | 2 | 136815.6 | 1 | 1 | 1 | 170042 | 0 |
| 28 | 15700772 | Nebechi | 571 | France | Male | 44 | 9 | 0 | 2 | 0 | 0 | 38433.35 | 0 |
| 29 | 15728693 | McWillian | 574 | Germany | Female | 43 | 3 | 141349.4 | 1 | 1 | 1 | 100187.4 | 0 |
| 30 | 15656300 | Lucciano | 411 | France | Male | 29 | 0 | 59697.17 | 2 | 1 | 1 | 53483.21 | 0 |
| 31 | 15589475 | Azikiwe | 591 | Spain | Female | 39 | 3 | 0 | 3 | 1 | 0 | 140469.4 | 1 |
| 32 | 15706552 | Odinakach | 533 | France | Male | 36 | 7 | 85311.7 | 1 | 0 | 1 | 156731.9 | 0 |
| 33 | 15750181 | Sanderson | 553 | Germany | Male | 41 | 9 | 110112.5 | 2 | 0 | 0 | 81898.81 | 0 |
| 34 | 15659428 | Maggard | 520 | Spain | Female | 42 | 6 | 0 | 2 | 1 | 1 | 34410.55 | 0 |
| 35 | 15732963 | Clements | 722 | Spain | Female | 29 | 9 | 0 | 2 | 1 | 1 | 142033.1 | 0 |

**5. Data Preprocessing:**

- This is the data cleaning and preparation phase:
- Handle missing data: Decide how to impute or remove missing values.
- Detect and deal with outliers: Outliers can adversely affect your model's performance.
- Encode categorical variables: Convert non-numeric data into numerical form, commonly through one-hot encoding.
- Scale or normalize numerical features: Ensure numerical attributes are on the same scale.

**SOURCE CODE**:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set_theme(color_codes=True)
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score,
classification_report, f1_score

dataset = pd.read_csv('/kaggle/input/churn-modelling/Churn_Modelling.csv')
dataset.head()
```

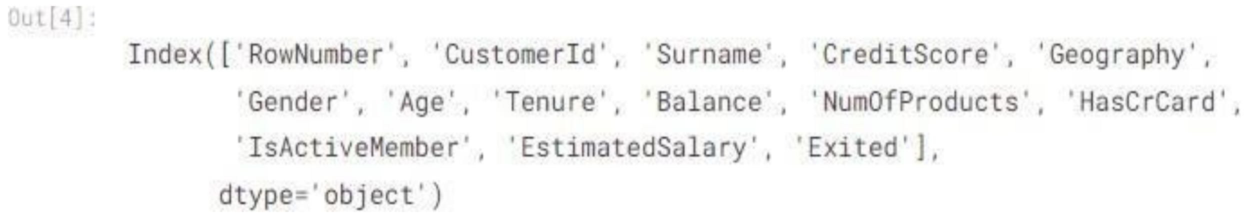


Out[3]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 |
| 1 | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 |
| 2 | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 |
| 3 | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 |
| 4 | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 |

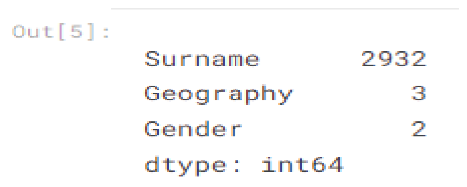
## Data Processing 1:

Dataset.columns



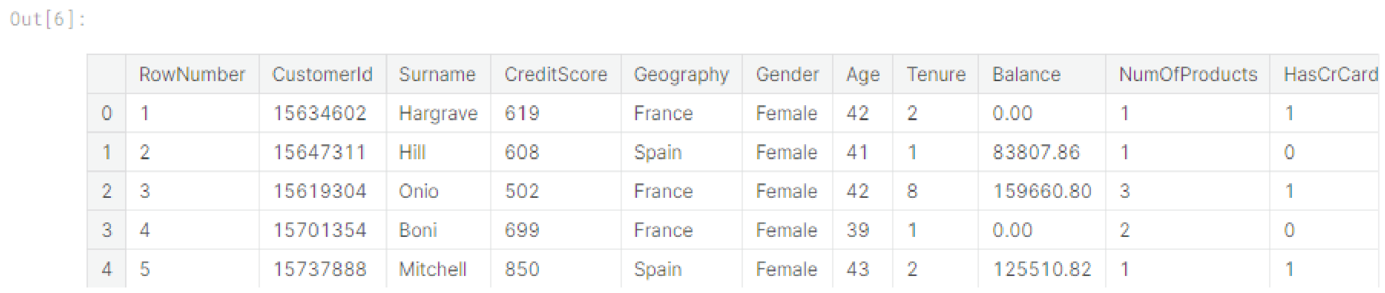
```
Out[4]:
        Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
               'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
               'IsActiveMember', 'EstimatedSalary', 'Exited'],
              dtype='object')
```

```
dataset.select_dtypes(include="object").nunique()
```



```
Out[5]:
        Surname      2932
        Geography       3
        Gender          2
        dtype: int64
```

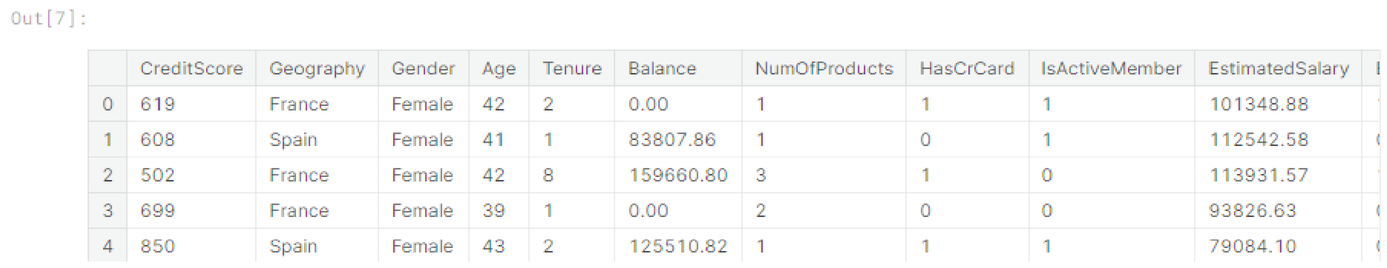
```
dataset.head()
```



Out[6]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 |
| 1 | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 |
| 2 | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 |
| 3 | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 |
| 4 | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 |

```
dataset.drop(columns = 'RowNumber',inplace = True)
dataset.drop(columns = 'CustomerId',inplace = True)
dataset.drop(columns = 'Surname',inplace = True)
dataset.head()
```



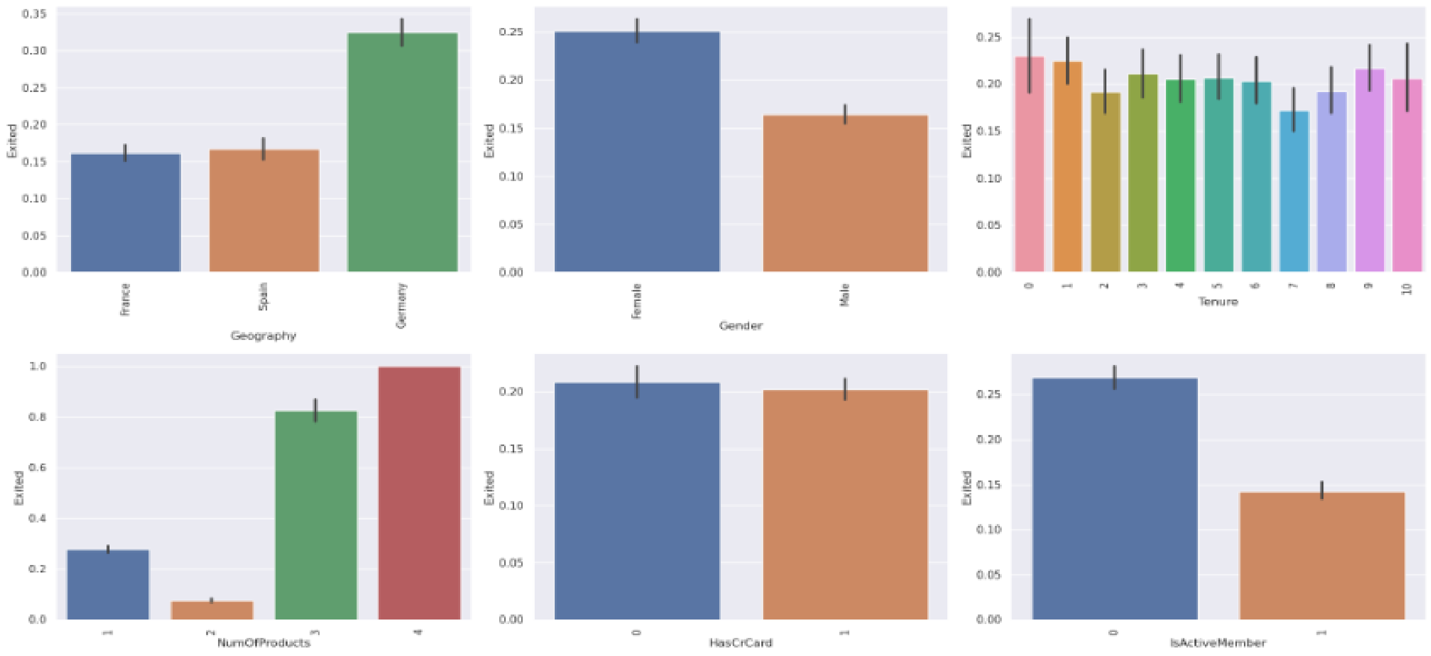
Out[7]:

| | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | 1 | 101348.88 |
| 1 | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 | 112542.58 |
| 2 | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | 0 | 113931.57 |
| 3 | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | 0 | 93826.63 |
| 4 | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | 1 | 79084.10 |

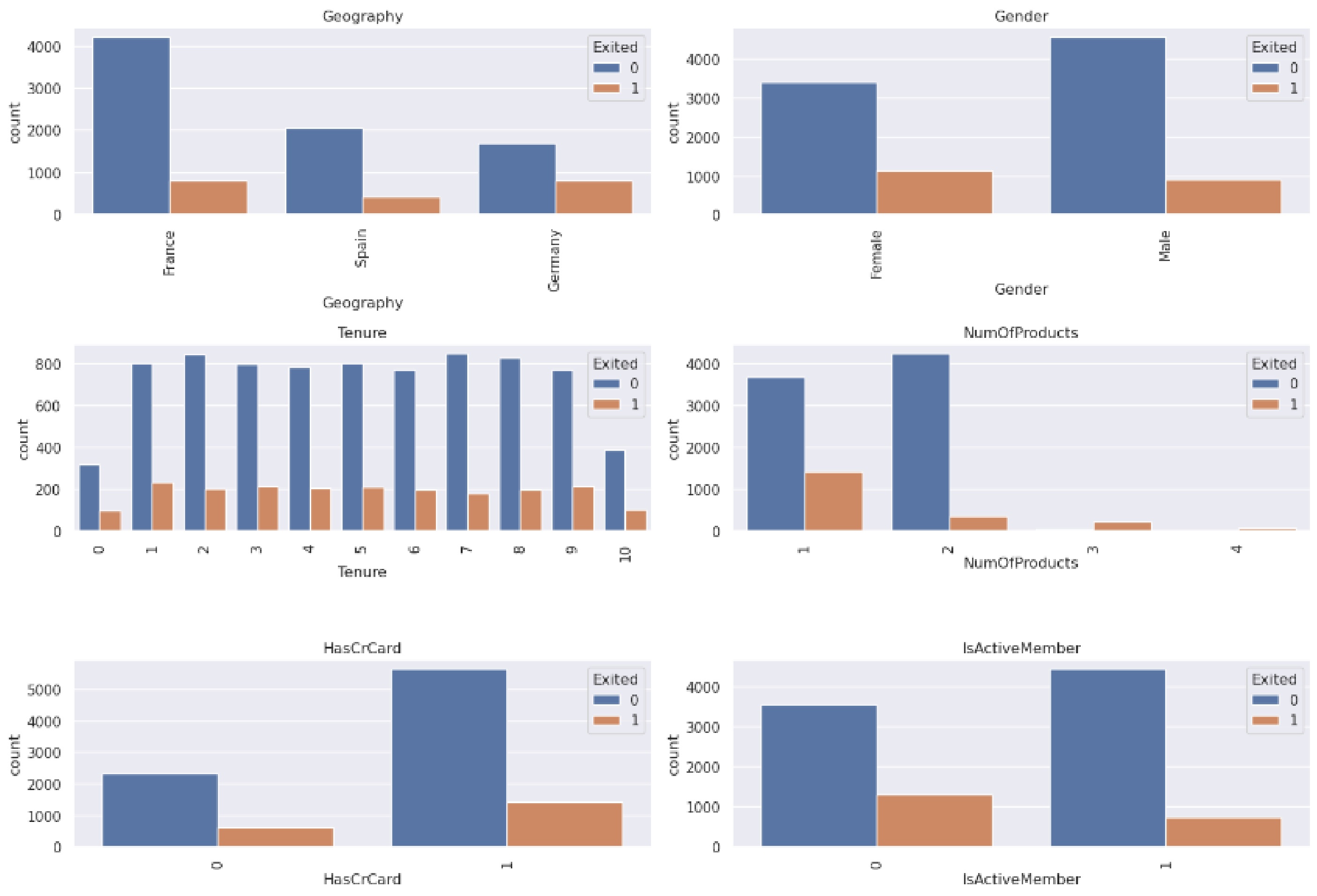
## Exploratory Data Analysis:

```
Cat_var = ['Geography','Gender','Tenure','NumOfProducts', 'HasCrCard','IsActiveMember']
fig, axs = plt.subplots(nrows= 2, ncols= 3,figsize= (20,10))
axs= axs.flatten()

for i,var in enumerate(Cat_var):
    sns.barplot(x=var,y='Exited',data=dataset ,ax=axs[i])
    axs[i].set_xticklabels(axs[i].get_xticklabels(), rotation = 90)
```

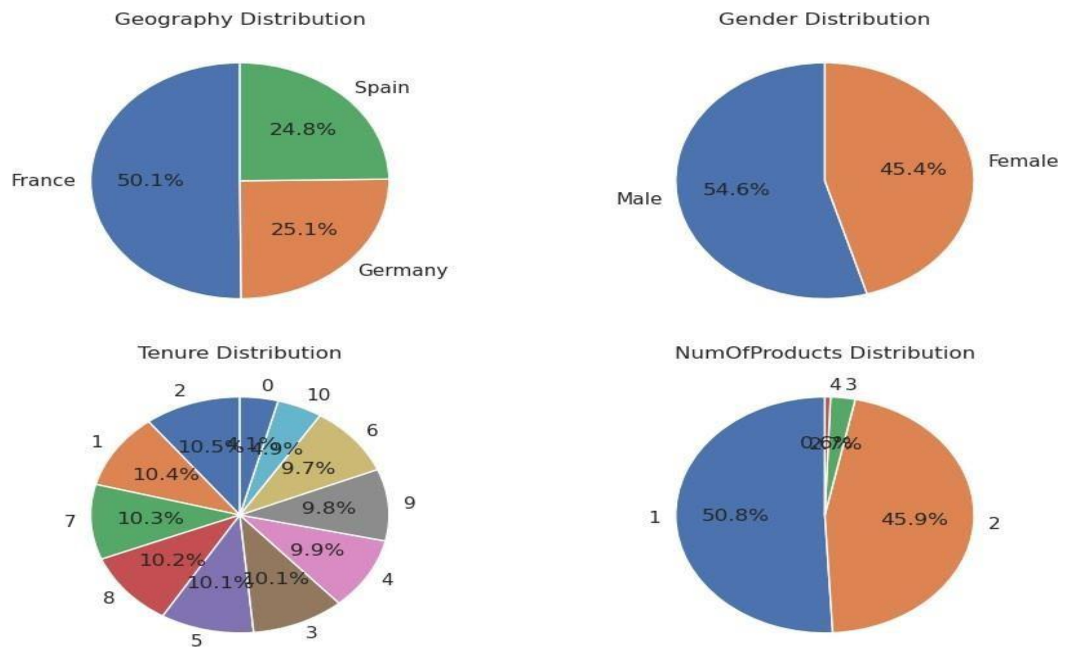
```
fig.tight_layout()
plt.show()
```



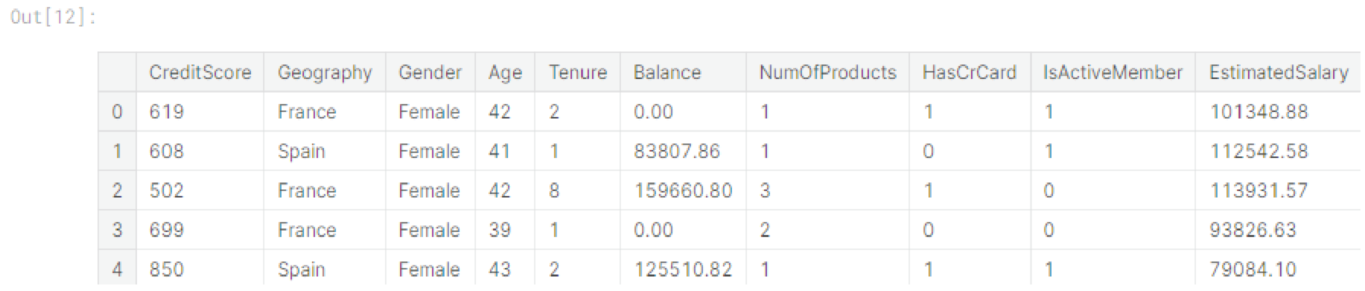
```
# Create a grid of subplots based on the number of categorical variables
num_cat_vars = len(Cat_var)
num_cols = 2  # You can adjust the number of columns as needed
num_rows = (num_cat_vars + num_cols - 1) // num_cols
fig, axs = plt.subplots(num_rows, num_cols, figsize=(15, 10))
axs = axs.flatten() # Flatten the 2D array of axes for easier indexing
for i, var in enumerate(Cat_var):
    row = i // num_cols
    col = i % num_cols
    ax = axs[i]

    sns.countplot(data=dataset, x=var, hue='Exited', ax=ax)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
    ax.set_title(var)
fig.tight_layout()
plt.show()
```



```
num_cat_vars = len(Cat_var)
num_cols = 2  # You can adjust the number of columns as needed
num_rows = (num_cat_vars + num_cols - 1) // num_cols
fig, axs = plt.subplots(num_rows, num_cols, figsize=(10, 10))
axs = axs.flatten()  # Flatten the 2D array of axes for easier indexing
for i, var in enumerate(Cat_var):
    if i < len(axs):
        ax = axs[i]
        cat_counts = dataset[var].value_counts()
        ax.pie(cat_counts, labels=cat_counts.index, autopct='%1.1f%%', startangle=90)
        ax.set_title(f'{var} Distribution')
fig.tight_layout()
if len(axs) > num_cat_vars:
    fig.delaxes(axs[-1])
plt.show()
```



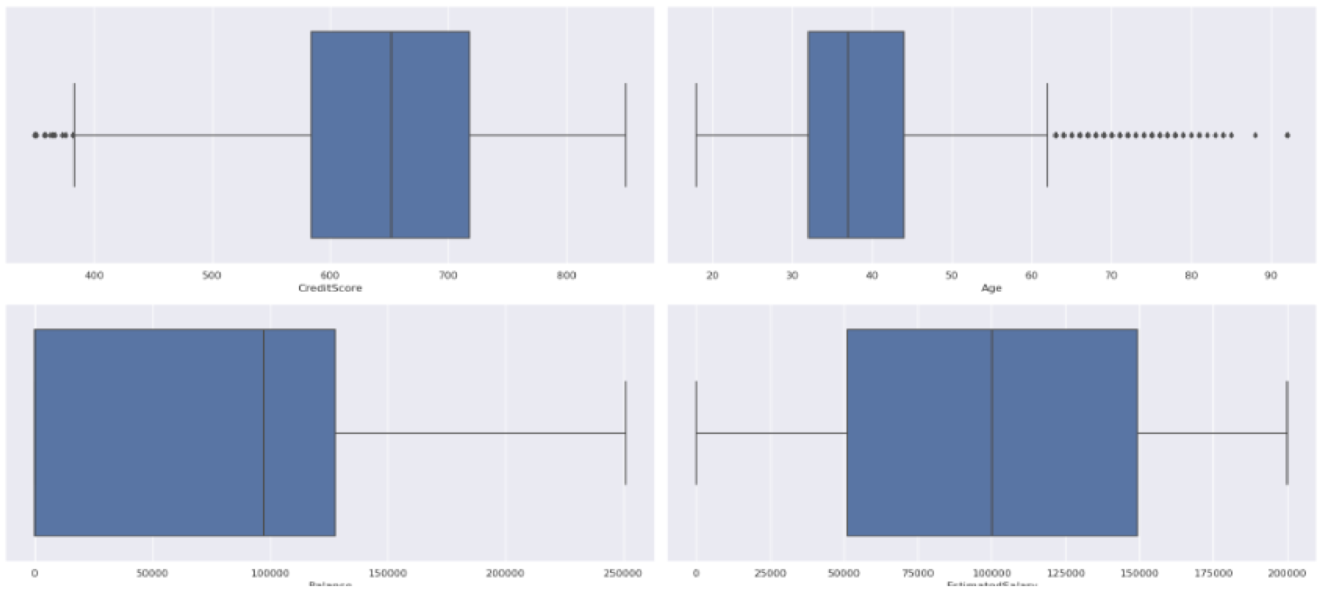
Dataset.heat()



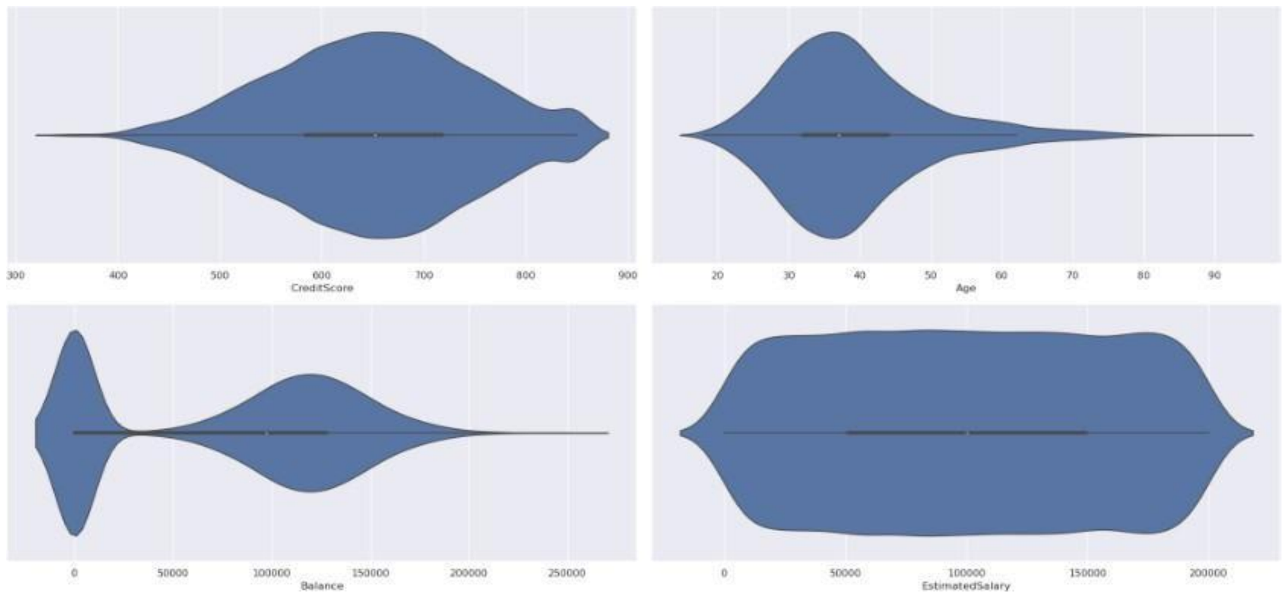
```
Out[12]:
```

| | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | 1 | 101348.88 |
| 1 | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 | 112542.58 |
| 2 | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | 0 | 113931.57 |
| 3 | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | 0 | 93826.63 |
| 4 | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | 1 | 79084.10 |

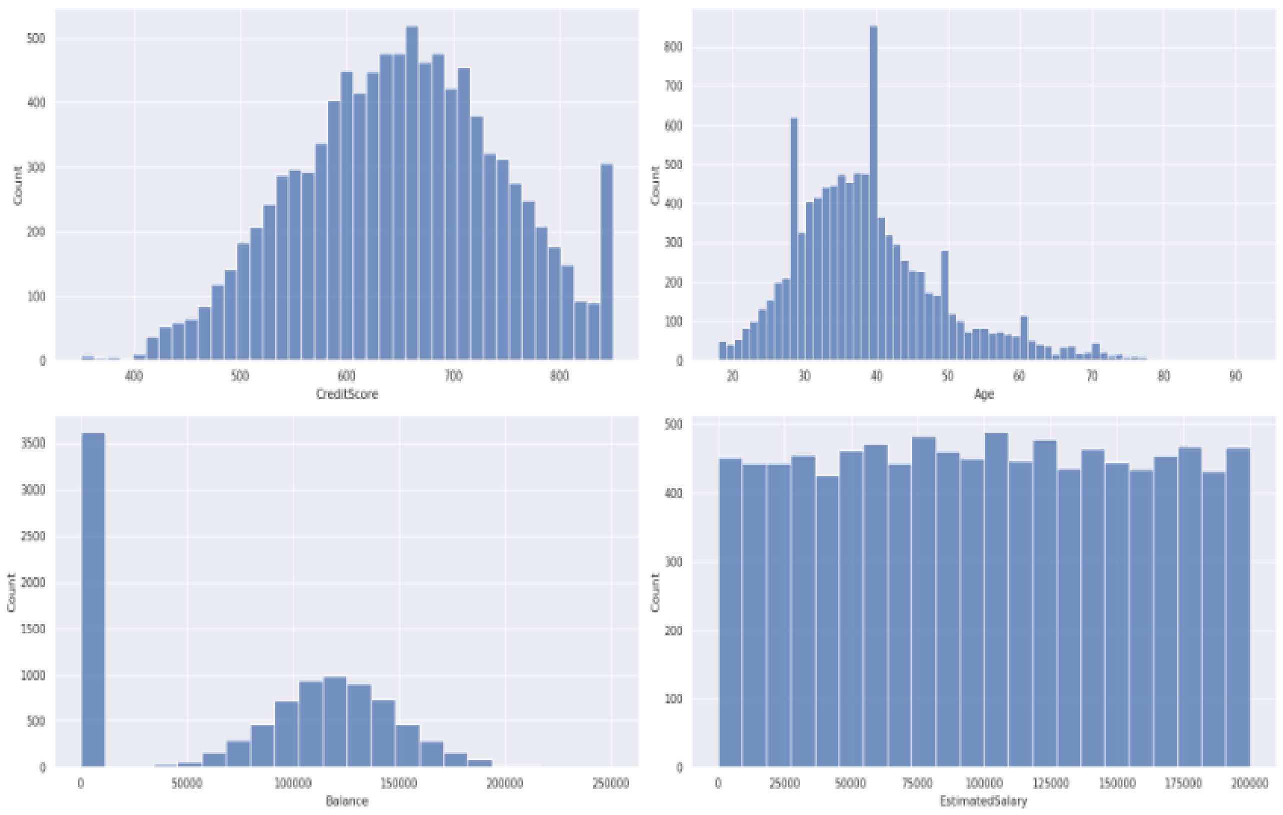
```
num_vars = ['CreditScore','Age','Balance','EstimatedSalary']
# Create a grid of subplots based on the number of categorical variables
num_cat_vars = len(num_vars)
num_cols = 2  # You can adjust the number of columns as needed
num_rows = (num_cat_vars + num_cols - 1) // num_cols
fig, axs = plt.subplots(num_rows, num_cols, figsize=(20, 10))
axs = axs.flatten()
for i, var in enumerate(num_vars):
    sns.boxplot(x=var,data=dataset, ax=axs[i])
# Adjust spacing between subplots
fig.tight_layout()
plt.show()
```



```
num_cat_vars = len(num_vars)
num_cols = 2  # You can adjust the number of columns as needed
num_rows = (num_cat_vars + num_cols - 1) // num_cols

fig, axs = plt.subplots(num_rows, num_cols, figsize=(20, 10))
axs = axs.flatten()

for i, var in enumerate(num_vars):
    sns.violinplot(x=var,data=dataset, ax=axs[i])
fig.tight_layout()


plt.show()
```



```
# Create a grid of subplots based on the number of categorical variables
num_cat_vars = len(num_vars)
num_cols = 2  # You can adjust the number of columns as needed

num_rows = (num_cat_vars + num_cols - 1) // num_cols
fig, axs = plt.subplots(num_rows, num_cols, figsize=(20, 10))
axs = axs.flatten()

for i, var in enumerate(num_vars):
    sns.histplot(x=var,data=dataset, ax=axs[i])
fig.tight_layout()
plt.show()
```



# SUBMISSION DOCUMENT

# DEVELOPMENT PART-2

## Introduction:

This document serves as a comprehensive submission for our project in IBM Watson Studio. In this submission, we will present the results of our data analysis, machine learning models, and other relevant findings. Our goal is to showcase the insights and solutions we have developed using IBM Watson Studio's powerful tools and capabilities. Please review this document to gain a clear understanding of our project's objectives, methodology, and outcomes.

IBM Watson Studio is an integrated development environment (IDE) designed to streamline the process of building, training, and deploying AI models and applications. It is part of IBM's suite of AI tools and services and is aimed at enabling data scientists, developers, and domain experts to collaborate effectively and efficiently in creating and deploying machine learning and deep learning models.

Machine learning model deployment is a critical phase in the machine learning lifecycle, involving the integration of trained models into real-world applications for making predictions or decisions based on new data. IBM Cloud Watson Studio provides a powerful and user-friendly platform for deploying machine learning models effectively and efficiently. Leveraging the comprehensive capabilities of IBM Cloud Watson Studio for model deployment enables organizations to scale AI applications and deliver actionable insights.

## 1. Prepare the Model:

Make sure your trained model is in a deployable format. It might be a Python script, a pre-trained machine learning model, or another suitable format.

## 2. Select a Relevant Dataset:

Carefully choose a dataset that aligns with your predictive use case. For customer churn prediction, your dataset might include information on customers, their interactions, demographics, purchase history, and whether they churned or not.

## 3. Set up Watson Studio:

Log in to your IBM Cloud account and access Watson Studio. This is your hub for all your machine learning activities.

## 4. Import the Dataset:

Use Watson Studio's data management tools to import your dataset. You can upload data files in various formats or connect to external data sources if your data is not already in Watson Studio.

| R.N | Cus.Id | Surname | C.Score | Geograph | Gender | Age | Tenure | Balance | N.pro | Hashcrd | A.Mem | Estimated | Exited |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0 | 1 | 1 | 1 | 101348.9 | 1 |
| 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 | 112542.6 | 0 |
| 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.8 | 3 | 1 | 0 | 113931.6 | 1 |
| 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0 | 2 | 0 | 0 | 93826.63 | 0 |
| 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.8 | 1 | 1 | 1 | 79084.1 | 0 |
| 6 | 15574012 | Chu | 645 | Spain | Male | 44 | 8 | 113755.8 | 2 | 1 | 0 | 149756.7 | 1 |
| 7 | 15592531 | Bartlett | 822 | France | Male | 50 | 7 | 0 | 2 | 1 | 1 | 10062.8 | 0 |
| 8 | 15656148 | Obinna | 376 | Germany | Female | 29 | 4 | 115046.7 | 4 | 1 | 0 | 119346.9 | 1 |
| 9 | 15792365 | He | 501 | France | Male | 44 | 4 | 142051.1 | 2 | 0 | 1 | 74940.5 | 0 |
| 10 | 15592389 | H? | 684 | France | Male | 27 | 2 | 134603.9 | 1 | 1 | 1 | 71725.73 | 0 |
| 11 | 15767821 | Bearce | 528 | France | Male | 31 | 6 | 102016.7 | 2 | 0 | 0 | 80181.12 | 0 |
| 12 | 15737173 | Andrews | 497 | Spain | Male | 24 | 3 | 0 | 2 | 1 | 0 | 76390.01 | 0 |
| 13 | 15632264 | Kay | 476 | France | Female | 34 | 10 | 0 | 2 | 1 | 0 | 26260.98 | 0 |
| 14 | 15691483 | Chin | 549 | France | Female | 25 | 5 | 0 | 2 | 0 | 0 | 190857.8 | 0 |
| 15 | 15600882 | Scott | 635 | Spain | Female | 35 | 7 | 0 | 2 | 1 | 1 | 65951.65 | 0 |
| 16 | 15643966 | Goforth | 616 | Germany | Male | 45 | 3 | 143129.4 | 2 | 0 | 1 | 64327.26 | 0 |
| 17 | 15737452 | Romeo | 653 | Germany | Male | 58 | 1 | 132602.9 | 1 | 1 | 0 | 5097.67 | 1 |
| 18 | 15788218 | Henderso | 549 | Spain | Female | 24 | 9 | 0 | 2 | 1 | 1 | 14406.41 | 0 |
| 19 | 15661507 | Muldrow | 587 | Spain | Male | 45 | 6 | 0 | 1 | 0 | 0 | 158684.8 | 0 |
| 20 | 15568982 | Hao | 726 | France | Female | 24 | 6 | 0 | 2 | 1 | 1 | 54724.03 | 0 |
| 21 | 15577657 | McDonald | 732 | France | Male | 41 | 8 | 0 | 2 | 1 | 1 | 170886.2 | 0 |
| 22 | 15597945 | Dellucci | 636 | Spain | Female | 32 | 8 | 0 | 2 | 1 | 0 | 138555.5 | 0 |
| 23 | 15699309 | Gerasimo | 510 | Spain | Female | 38 | 4 | 0 | 1 | 1 | 0 | 118913.5 | 1 |
| 24 | 15725737 | Mosman | 669 | France | Male | 46 | 3 | 0 | 2 | 0 | 1 | 8487.75 | 0 |
| 25 | 15625047 | Yen | 846 | France | Female | 38 | 5 | 0 | 1 | 1 | 1 | 187616.2 | 0 |
| 26 | 15738191 | Maclean | 577 | France | Male | 25 | 3 | 0 | 2 | 0 | 1 | 124508.3 | 0 |
| 27 | 15736816 | Young | 756 | Germany | Male | 36 | 2 | 136815.6 | 1 | 1 | 1 | 170042 | 0 |
| 28 | 15700772 | Nebechi | 571 | France | Male | 44 | 9 | 0 | 2 | 0 | 0 | 38433.35 | 0 |
| 29 | 15728693 | McWillian | 574 | Germany | Female | 43 | 3 | 141349.4 | 1 | 1 | 1 | 100187.4 | 0 |
| 30 | 15656300 | Lucciano | 411 | France | Male | 29 | 0 | 59697.17 | 2 | 1 | 1 | 53483.21 | 0 |
| 31 | 15589475 | Azikiwe | 591 | Spain | Female | 39 | 3 | 0 | 3 | 1 | 0 | 140469.4 | 1 |
| 32 | 15706552 | Odinakach | 533 | France | Male | 36 | 7 | 85311.7 | 1 | 0 | 1 | 156731.9 | 0 |
| 33 | 15750181 | Sandersor | 553 | Germany | Male | 41 | 9 | 110112.5 | 2 | 0 | 0 | 81898.81 | 0 |
| 34 | 15659428 | Maggard | 520 | Spain | Female | 42 | 6 | 0 | 2 | 1 | 1 | 34410.55 | 0 |
| 35 | 15732963 | Clements | 722 | Spain | Female | 29 | 9 | 0 | 2 | 1 | 1 | 142033.1 | 0 |

5. **Data Preprocessing:**
- This is the data cleaning and preparation phase:
- Handle missing data: Decide how to impute or remove missing values.
- Detect and deal with outliers: Outliers can adversely affect your model's performance.
- Encode categorical variables: Convert non-numeric data into numerical form, commonly through one-hot encoding.
- Scale or normalize numerical features: Ensure numerical attributes are on the same scale.

6. **Feature Selection**:
- Identify which features (attributes) are most relevant to your predictive model. Use Watson Studio's tools for feature selection or explore feature importance metrics.

7. **Model Training**:
- Select an appropriate machine learning algorithm for your use case:
- Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, Neural Networks, etc.
- Split your dataset into training and testing sets to assess your model's performance.

8. **Model Evaluation:**
- Assess how well your model performs using appropriate metrics: - Accuracy, Precision, Recall, F1-score, ROC-AUC, etc.
- Watson Studio provides tools to help you with model evaluation.

9. **Fine-Tuning:**
- If your model's performance is unsatisfactory, you can:
- Adjust hyperparameters: Tweak the settings of your algorithm.
- Try different algorithms: Experiment with different machine learning techniques.
- Iterate on the training process to improve results.

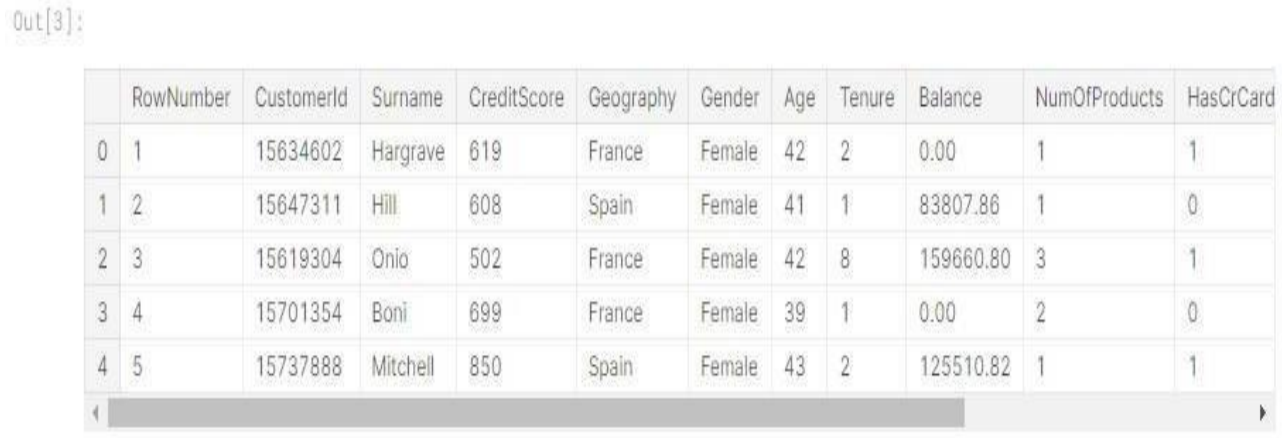
10. **Deployment:**
- Once you're content with your model's performance, you can deploy it using Watson Studio.
- Deployed models can be used for making predictions, integrated into applications, and accessed through APIs for real-time use.

**SOURCE CODE**:

```
import pandas as pd import
matplotlib.pyplot as plt import
seaborn as sns import numpy
as np
```

sns.set_theme(color_codes=Tr
ue)
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, classification_report, f1_score

dataset = pd.read_csv('/kaggle/input/churn-modelling/Churn_Modelling.csv') dataset.head()

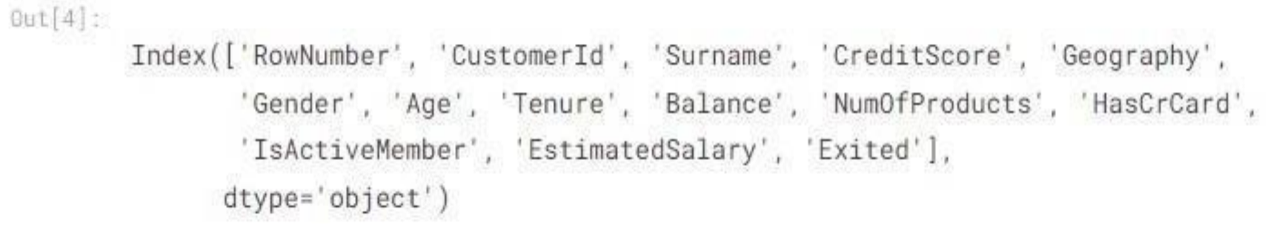


Out[3]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 |
| 1 | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 |
| 2 | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 |
| 3 | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 |
| 4 | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 |

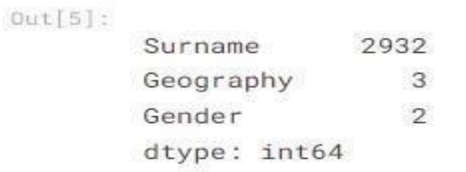
## Data Processing 1:

Dataset.columns



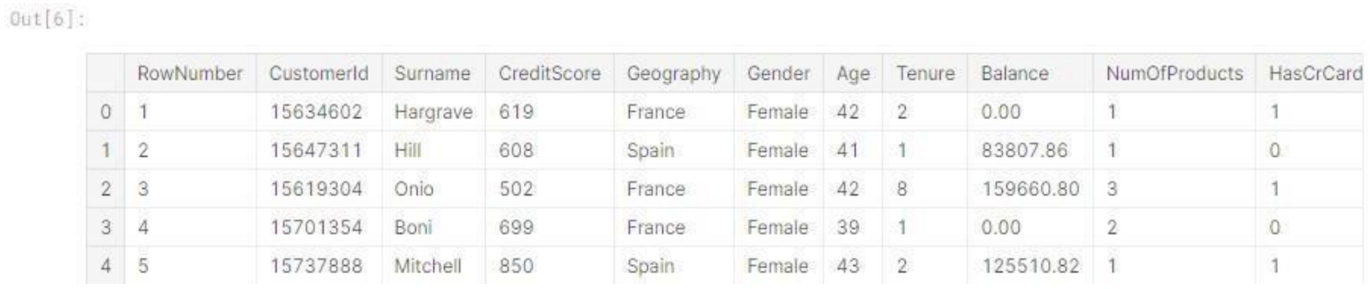
```
Out[4]:
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

dataset.select_dtypes(include="object").nunique()



```
Out[5]:
Surname      2932
Geography       3
Gender          2
dtype: int64
```

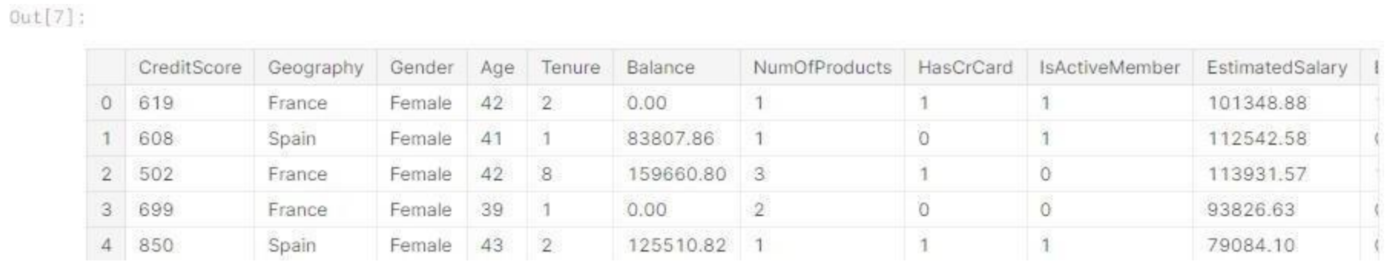
dataset.head()



Out[6]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 |
| 1 | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 |
| 2 | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 |
| 3 | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 |
| 4 | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 |

```
dataset.drop(columns = 'RowNumber',inplace = True)
dataset.drop(columns = 'CustomerId',inplace = True)
dataset.drop(columns = 'Surname',inplace = True) dataset.head()
```

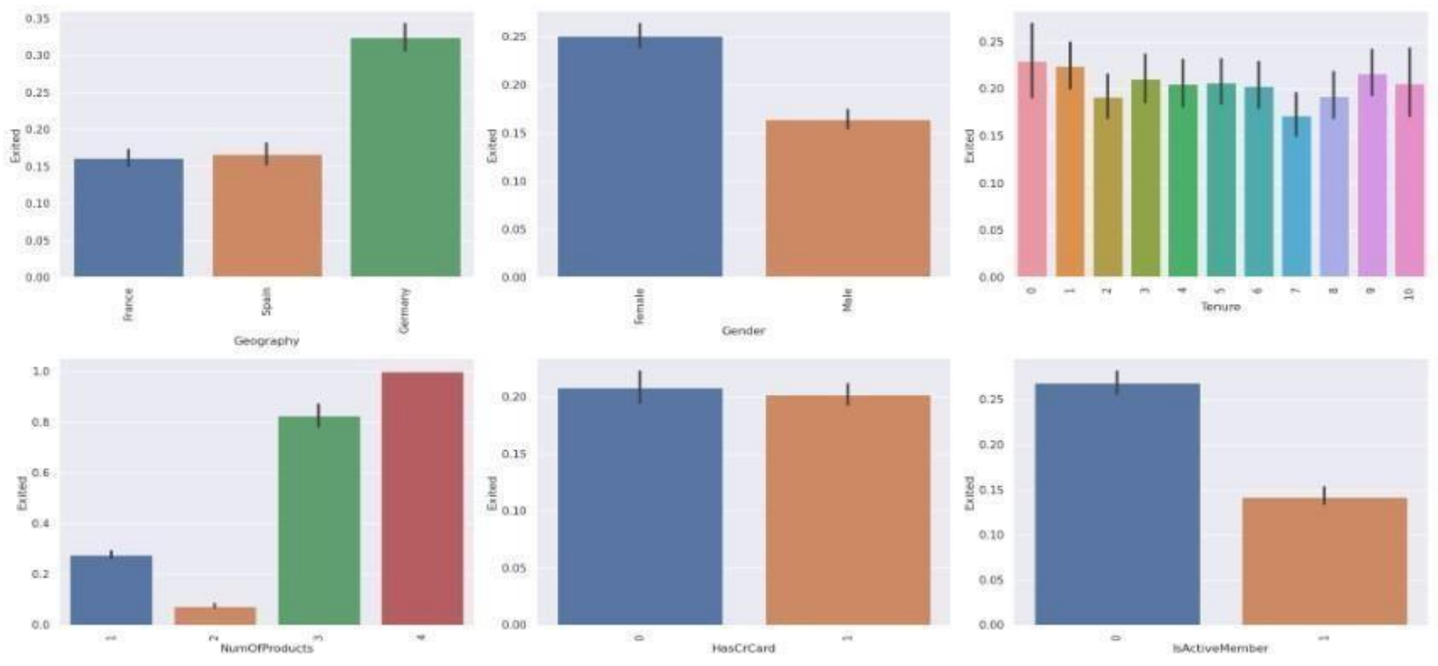


```
Out[7]:
```

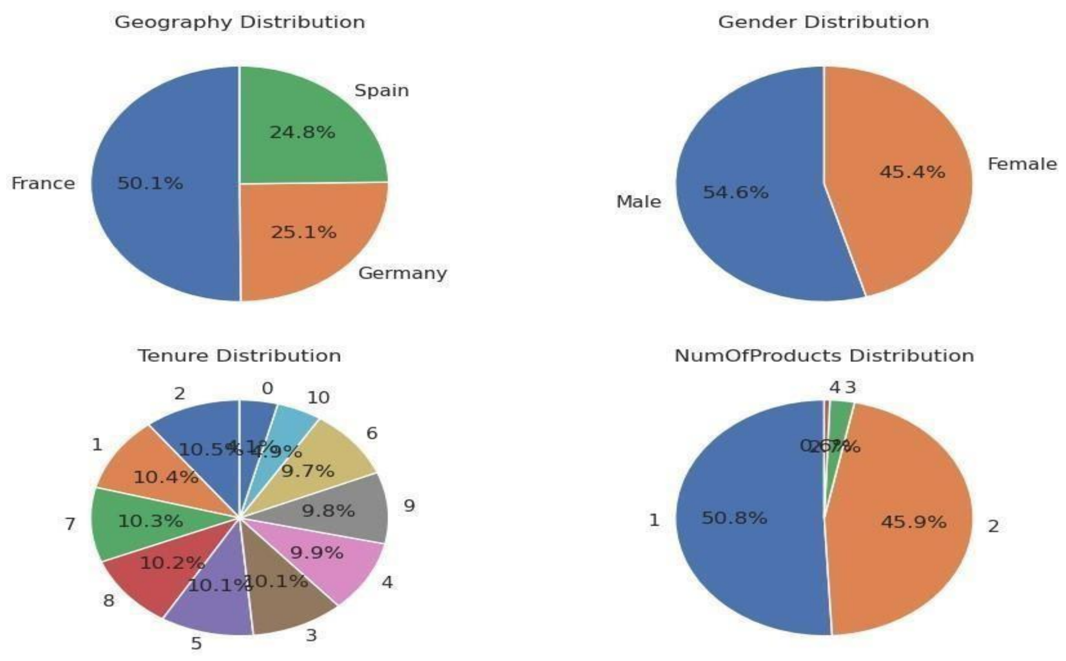
| | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | 1 | 101348.88 |
| 1 | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 | 112542.58 |
| 2 | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | 0 | 113931.57 |
| 3 | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | 0 | 93826.63 |
| 4 | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | 1 | 79084.10 |

## Exploratory Data Analysis:

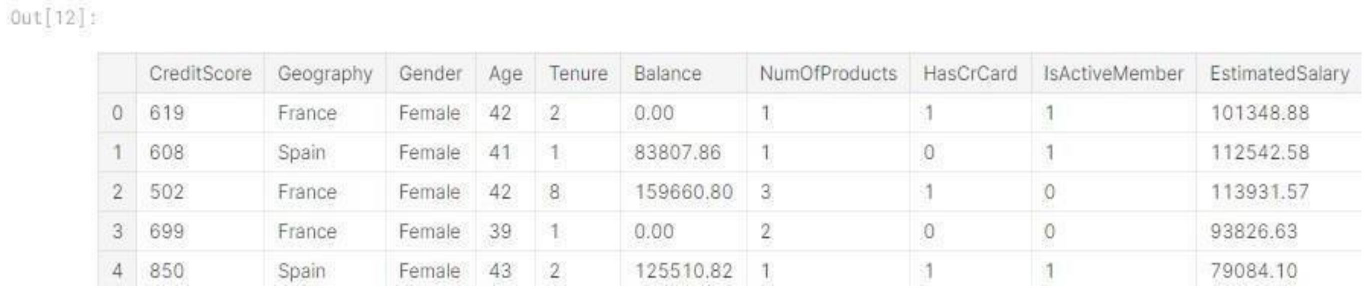
```
Cat_var = ['Geography','Gender','Tenure','NumOfProducts', 'HasCrCard','IsActiveMember']
fig, axs = plt.subplots(nrows= 2, ncols= 3,figsize= (20,10)) axs= axs.flatten() for i,var in
enumerate(Cat_var):      sns.barplot(x=var,y='Exited',data=dataset ,ax=axs[i])
axs[i].set_xticklabels(axs[i].get_xticklabels(), rotation = 90) fig.tight_layout()
plt.show()
```



```
num_cat_vars = len(Cat_var)
num_cols = 2 # You can adjust the number of columns as needed
num_rows = (num_cat_vars + num_cols - 1) // num_cols fig, axs =
plt.subplots(num_rows, num_cols, figsize=(10, 10)) axs =
axs.flatten()  # Flatten the 2D array of axes for easier indexing for i,
var in enumerate(Cat_var):      if i < len(axs):          ax = axs[i]
        cat_counts = dataset[var].value_counts()
        ax.pie(cat_counts, labels=cat_counts.index, autopct='%1.1f%%', startangle=90)
ax.set_title(f'{var} Distribution')
fig.tight_layout() if
len(axs) > num_cat_vars:
    fig.delaxes(axs[-1])
plt.show()
```



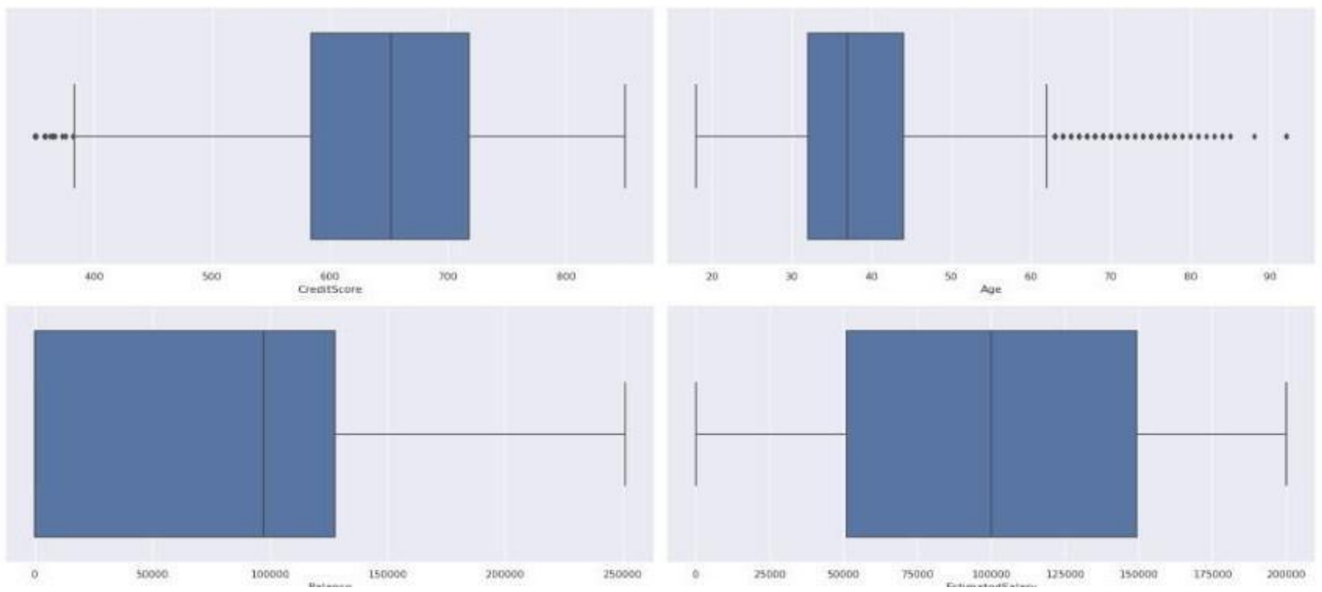
Dataset.heat()



```
Out[12]:
```

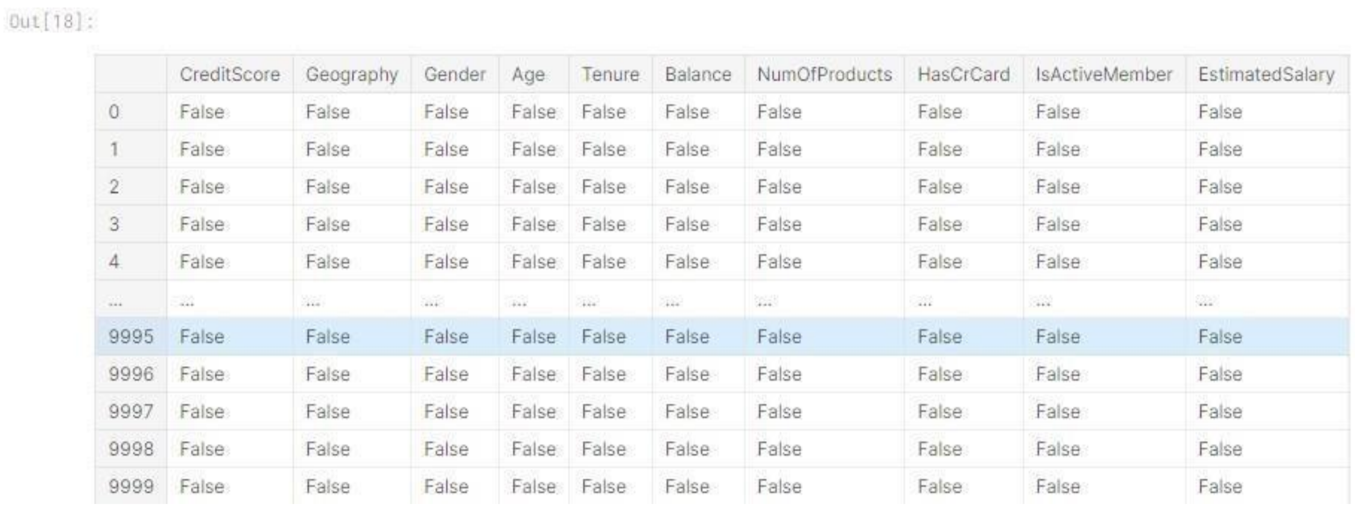
| | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | 1 | 101348.88 |
| 1 | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 | 112542.58 |
| 2 | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | 0 | 113931.57 |
| 3 | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | 0 | 93826.63 |
| 4 | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | 1 | 79084.10 |

```
num_vars = ['CreditScore','Age','Balance','EstimatedSalary']
# Create a grid of subplots based on the number of categorical variables num_cat_vars
= len(num_vars)
num_cols = 2  # You can adjust the number of columns as needed
num_rows = (num_cat_vars + num_cols - 1) // num_cols fig, axs
= plt.subplots(num_rows, num_cols, figsize=(20, 10)) axs =
axs.flatten() for i, var in enumerate(num_vars):
sns.boxplot(x=var,data=dataset, ax=axs[i])
# Adjust spacing between subplots
fig.tight_layout()
plt.show()
```



## Data Processing 2:

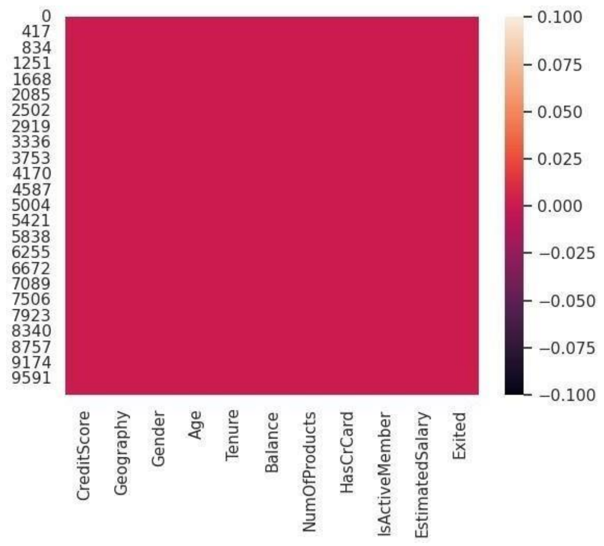
```
dataset.isnull()
```



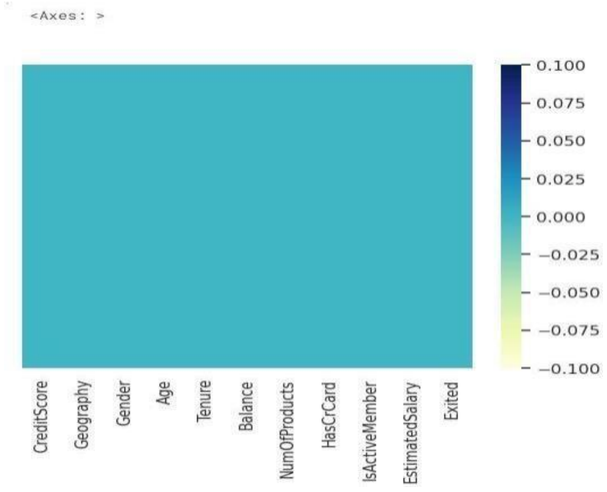
Out[18]:

| | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | False | False | False | False | False | False | False | False | False | False |
| 1 | False | False | False | False | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False | False | False | False | False |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9995 | False | False | False | False | False | False | False | False | False | False |
| 9996 | False | False | False | False | False | False | False | False | False | False |
| 9997 | False | False | False | False | False | False | False | False | False | False |
| 9998 | False | False | False | False | False | False | False | False | False | False |
| 9999 | False | False | False | False | False | False | False | False | False | False |

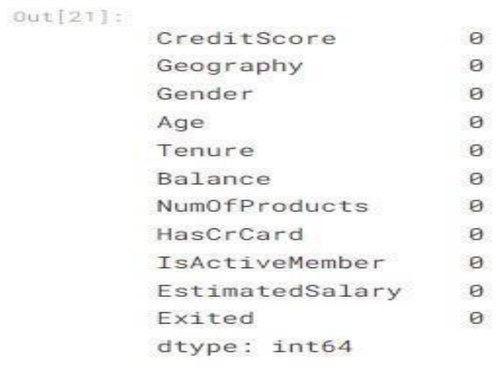
sns.heatmap(dataset.isnull())



sns.heatmap(dataset.isnull() ,yticklabels=False , cmap="YlGnBu")

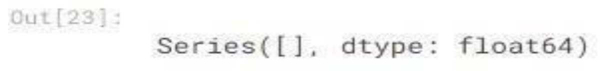


*dataset.isnull().sum()*



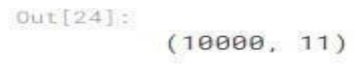
```
Out[21]:
         CreditScore        0
         Geography          0
         Gender             0
         Age                0
         Tenure             0
         Balance            0
         NumOfProducts      0
         HasCrCard          0
         IsActiveMember     0
         EstimatedSalary    0
         Exited             0
         dtype: int64
```

```
cheak_missing = dataset.isnull().sum() * 100/ dataset.shape[0]
cheak_missing[cheak_missing > 0].sort_values(ascending=False)
```



```
Out[23]:
        Series([], dtype: float64)
```

```
dataset.shape
```



```
Out[24]:
        (10000, 11)
```

## Label Encoding:

```
for col in dataset.select_dtypes(include=['object']).columns:
   #print column name and value
print(f"{col}:{dataset[col].unique()}")
```

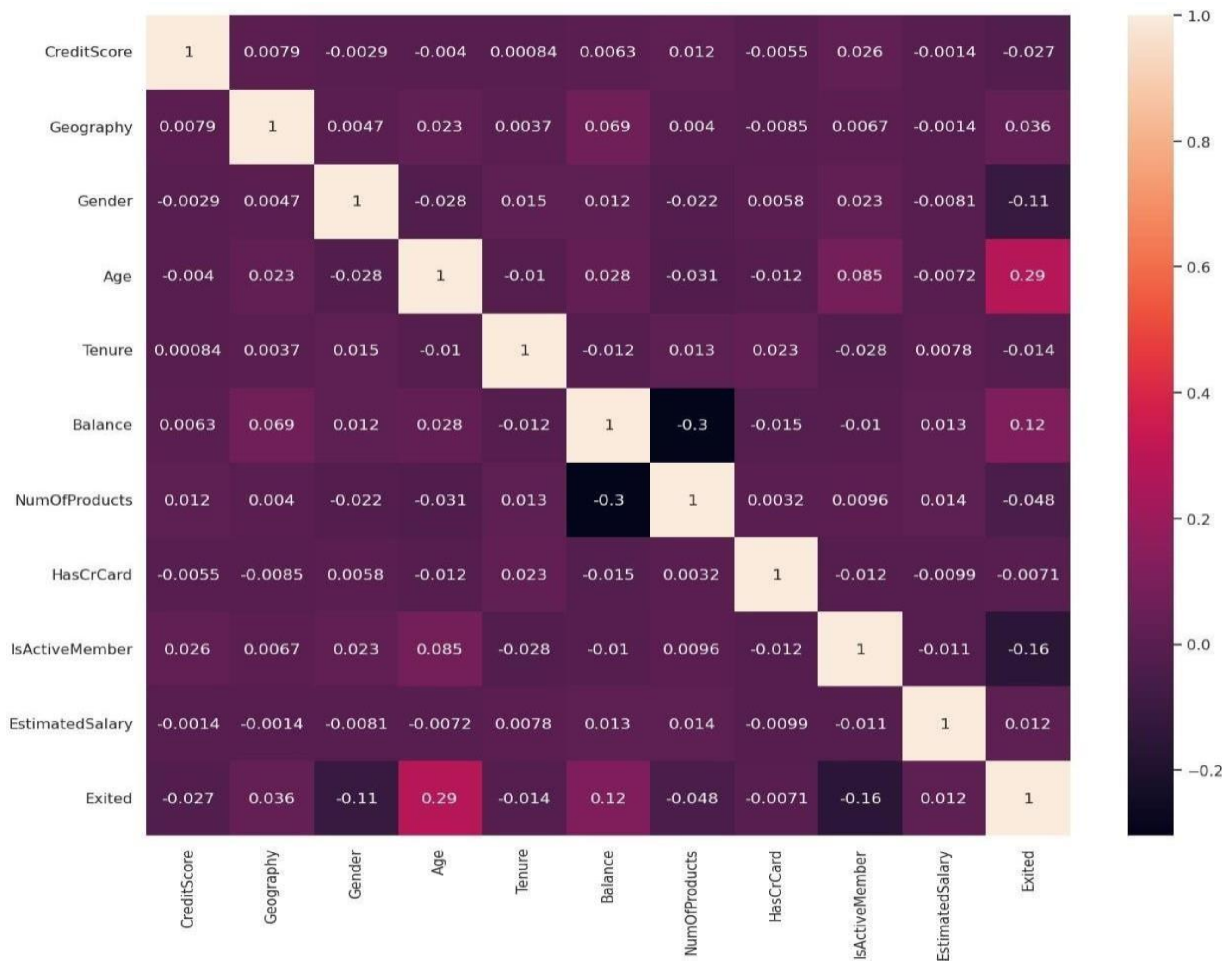
```
Geography:['France' 'Spain' 'Germany']
Gender:['Female' 'Male']
```

```
from sklearn import preprocessing

#loop to find object datatype for col in
dataset.select_dtypes(include=['object']).columns:

#initilization of LabelEncoder
   label_encoding= preprocessing.LabelEncoder()

label_encoding.fit(dataset[col].unique())      dataset[col]
= label_encoding.transform(dataset[col])
print(f"{col}:{dataset[col].unique()}")
```

```
Geography:[0 2 1]
Gender:[0 1]
```

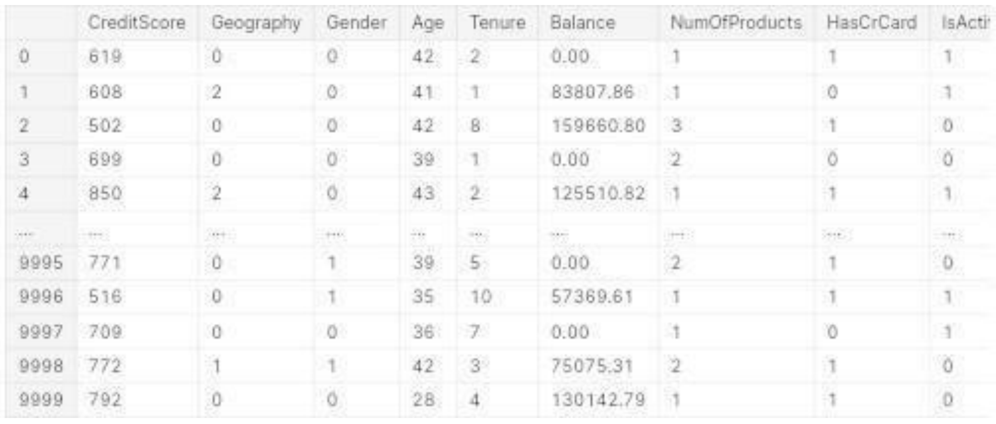
## Removing Outliers using IQR: Heatmap Correlation:

```
plt.figure(figsize=(15,12))
sns.heatmap(dataset.corr(),fmt=’.2g’,annot=true)
```



| | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary | Exited |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CreditScore | 1 | 0.0079 | -0.0029 | -0.004 | 0.00084 | 0.0063 | 0.012 | -0.0055 | 0.026 | -0.0014 | -0.027 |
| Geography | 0.0079 | 1 | 0.0047 | 0.023 | 0.0037 | 0.069 | 0.004 | -0.0085 | 0.0067 | -0.0014 | 0.036 |
| Gender | -0.0029 | 0.0047 | 1 | -0.028 | 0.015 | 0.012 | -0.022 | 0.0058 | 0.023 | -0.0081 | -0.11 |
| Age | -0.004 | 0.023 | -0.028 | 1 | -0.01 | 0.028 | -0.031 | -0.012 | 0.085 | -0.0072 | 0.29 |
| Tenure | 0.00084 | 0.0037 | 0.015 | -0.01 | 1 | -0.012 | 0.013 | 0.023 | -0.028 | 0.0078 | -0.014 |
| Balance | 0.0063 | 0.069 | 0.012 | 0.028 | -0.012 | 1 | -0.3 | -0.015 | -0.01 | 0.013 | 0.12 |
| NumOfProducts | 0.012 | 0.004 | -0.022 | -0.031 | 0.013 | -0.3 | 1 | 0.0032 | 0.0096 | 0.014 | -0.048 |
| HasCrCard | -0.0055 | -0.0085 | 0.0058 | -0.012 | 0.023 | -0.015 | 0.0032 | 1 | -0.012 | -0.0099 | -0.0071 |
| IsActiveMember | 0.026 | 0.0067 | 0.023 | 0.085 | -0.028 | -0.01 | 0.0096 | -0.012 | 1 | -0.011 | -0.16 |
| EstimatedSalary | -0.0014 | -0.0014 | -0.0081 | -0.0072 | 0.0078 | 0.013 | 0.014 | -0.0099 | -0.011 | 1 | 0.012 |
| Exited | -0.027 | 0.036 | -0.11 | 0.29 | -0.014 | 0.12 | -0.048 | -0.0071 | -0.16 | 0.012 | 1 |

Dataset



| | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActi |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 619 | 0 | 0 | 42 | 2 | 0.00 | 1 | 1 | 1 |
| 1 | 608 | 2 | 0 | 41 | 1 | 83807.86 | 1 | 0 | 1 |
| 2 | 502 | 0 | 0 | 42 | 8 | 159660.80 | 3 | 1 | 0 |
| 3 | 699 | 0 | 0 | 39 | 1 | 0.00 | 2 | 0 | 0 |
| 4 | 850 | 2 | 0 | 43 | 2 | 125510.82 | 1 | 1 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9995 | 771 | 0 | 1 | 39 | 5 | 0.00 | 2 | 1 | 0 |
| 9996 | 516 | 0 | 1 | 35 | 10 | 57369.61 | 1 | 1 | 1 |
| 9997 | 709 | 0 | 0 | 36 | 7 | 0.00 | 1 | 0 | 1 |
| 9998 | 772 | 1 | 1 | 42 | 3 | 75075.31 | 2 | 1 | 0 |
| 9999 | 792 | 0 | 0 | 28 | 4 | 130142.79 | 1 | 1 | 0 |

```
x=dataset.drop(“Exited”,axis=1)
y=dataset[‘Exited’]

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,
              test_size=0.20,random_state=42)
```

```
from keras.models import Sequential
```

```
/opt/conda/lib/python3.10/site-packages/tensorflow_io/python/ops/_init_.py:98: UserWarning:
unable to loadlibtensorflow_io_plugins.so: unable to open file: libtensorflow_io_plugins.so,
from paths: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io_plugins.so']
caused by: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io_plugins.so: undefined symbol:
_ZN3tsl6StatusC1EN10tensorflow5error4CodeESt17basic_string_viewIcSt11char_traitsIcEENS_14
SourceLocatio nE']
  warnings.warn(f"unable to load libtensorflow_io_plugins.so: {e}")
/opt/conda/lib/python3.10/site-packages/tensorflow_io/python/ops/_init_.py:104: UserWarning:
file systemplugins are not loaded: unable to open file: libtensorflow_io.so, from paths:
['/opt/conda/lib/python3.10/site- packages/tensorflow_io/python/ops/libtensorflow_io.so']
caused by: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io.so: undefinedsymbol:
_ZTVN10tensorflow13GcsFileSystemE']
  warnings.warn(f"file system plugins are not loaded: {e}")
```

## **CONCLUSION**:

In conclusion, IBM Cloud Watson Studio serves as an indispensable tool for enterprises and data professionals embarking on the journey of machine learning model development. By offering a seamless and collaborative workspace, Watson Studio enables streamlined data exploration, efficient model training, and simplified deployment processes.

With its emphasis on scalability and integration with IBM Cloud services, Watson Studio empowers businesses to unlock the full potential of their data, enabling them to make data-driven decisions and stay ahead in an increasingly competitive landscape. By leveraging the capabilities of Watson Studio, organizations can drive innovation, enhance operational efficiency, and pave the way for future advancements in the field of artificial intelligence and machine learning.

Machine learning model development with IBM Cloud Watson Studio offers a comprehensive and efficient solution for organizations and individuals seeking to harness the full potential of artificial intelligence. Throughout this guide, we’ve explored the rich ecosystem of tools and resources that Watson Studio provides, allowing data scientists and developers to seamlessly navigate the complex landscape of data-driven decision-making.

With Watson Studio, you gain access to robust data management, collaborative workspaces, and a wide array of machine learning algorithms. This platform simplifies and accelerates the entire machine learning lifecycle, from data preparation and feature engineering to model development and deployment. The integration with IBM Watson AI services opens doors to advanced AI capabilities, making it easier than ever to infuse intelligence into your applications.

Moreover, Watson Studio is part of the IBM Cloud ecosystem, offering scalability,security, and the flexibility to adapt to your organization’s needs. Whether you’re working on small-scale projects or enterprise-level initiatives, Watson Studio provides the tools and support required to drive innovation and stay competitive in today’s data-driven world.
In summary, IBM Cloud Watson Studio empowers you to turn data into actionable insights, make predictions, and build intelligent applications with confidence.

It’s a platform designed to foster collaboration, accelerate development, and unlock the full potential of machine learning, making it an invaluable asset for anyone on the journey to harness the power of artificial intelligence.