**Phase 5**

**PROJECT DOCUMENTATION & SUBMISSION**

|  |  |
| --- | --- |
| **Date** | **31-10-2023** |
| **Team ID** | **671** |
| **Project Name** | **Machine Learning Model Deployment using IBM Cloud Watson Studio** |

**Project Title: Phishing Detection Model Deployment in IBM Watson Studio**

**Problem Statement**

Objective: Develop a robust machine learning model deployment pipeline for phishing detection with IBM Cloud Watson Studio.

**Problem identified:**

Online security is an increasingly critical concern in our digital age. Phishing attacks, in which cybercriminals impersonate trusted entities to deceive individuals and organizations, continue to evolve and threaten the safety of online interactions. Recognizing these threats in real-time is challenging, and the need for robust and efficient phishing detection solutions is more pressing than ever. While many techniques exist, deploying a reliable and accessible system for phishing detection remains a complex task. Our problem statement addresses this need.

**Introduction:**

In response to this pressing issue, our project, "Machine Learning Model Deployment using IBM Cloud Watson Studio," aims to deploy an innovative and practical solution for real-time phishing detection. The project leverages IBM Watson Studio, a powerful machine learning platform, to create, deploy, and maintain a cutting-edge machine learning model.

This project begins by addressing the core issue: recognizing phishing websites and emails promptly. It is a challenging problem, as malicious actors constantly adapt their tactics. Traditional rule-based methods are no longer sufficient. Instead, we turn to machine learning, a field that offers the promise of adaptability and scalability.

Our project focuses on a well-structured process that spans from data collection and preprocessing to model selection and deployment. We employ the AutoAI feature in IBM Watson Studio to streamline model creation and hyperparameter tuning. The ultimate goal is to create a system that is accessible to users and applications, providing real-time predictions through a secure API endpoint.

In the following sections, we will delve into the intricacies of our project, describing the methods, tools, and datasets we utilize to build a robust phishing detection system. We aim to address a pressing issue in online security, empowering individuals and organizations to protect themselves in an ever-evolving digital landscape.

**Data:** We possess a dataset comprising various email attributes, URLs, and user behaviours, along with their classification as either phishing or legitimate. This dataset will be instrumental in training and evaluating our machine learning model.

**LITERATURE SURVEY**

**1. “Implementation of Data Mining on a Secure Cloud Computing via Web API Using Supervised Machine Learning Algorithm”, Tosin Ige [2022]**

This research paper focuses on the implementation of data mining in a secure cloud computing environment through a combination of decision tree and Random Forest algorithms, accessible via a Restful Application Programming Interface (API). It addresses the challenge of efficiently and securely mining large volumes of data available through cloud computing for pattern detection. The study bypasses direct interaction with data warehouses to ensure security and scalability, using a combination of IBM Cloud storage, a web service, an API, and a decision tree/Random Forest algorithm. The achieved model exhibits a high accuracy rate of 94%.

**2. “Machine Learning and IBM Cloud for Critical Patient Care: A Promising Solution”, Asif Ahmed Neloy [2019]**

This research project, conducted by a team from North South University in Dhaka, Bangladesh, aims to revolutionize critical patient care in healthcare facilities. The team proposes the development of a comprehensive system that harnesses the power of machine learning (ML) and the IBM Cloud platform. The key objective is to enable real-time monitoring and prediction of critical patients' health conditions, allowing doctors and nurses to provide timely care. The project involves the use of various ML algorithms, ensemble methods, and the creation of a mobile application named "Critical Patient Management System - CPMS" for seamless data access. By integrating advanced technology, this project seeks to address the critical patient care challenges prevalent in developing countries like Bangladesh.

**3. “The Development and Deployment of Machine Learning Models”, James A. Pruneski [2022]**

The application of artificial intelligence, particularly machine learning, is gaining traction in Orthopaedic Surgery and the field of medicine at large. This growing interest is shared by data scientists and physicians, although there is often a gap in understanding the developmental process and potential applications of machine learning. Given the anticipated impact of new technology on clinical practice in the coming years, it is crucial for physicians to grasp the workings of these processes. This paper aims to provide clarity and a general framework for building and evaluating machine learning models.

**4. “Explainable Machine Learning in Deployment” ,Shubham Sharma [2020]**

This study, conducted by Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley, delves into the practical deployment of explainable machine learning (ML). While explainability is crucial for building trust in ML models, the research aims to understand how organizations use explainability techniques in practice. The study reveals that most deployments primarily serve machine learning engineers for model debugging, rather than end users affected by the models, creating a gap between explainability in practice and transparency goals. It highlights the limitations of current explainability techniques and proposes a framework for setting clear goals for explainability to facilitate end-user interaction.

**5. “Accelerating the Machine Learning Model Deployment using MLOps”, Mandepudi Nobel Chowdary[2022]**

This research, led by Mandepudi Nobel Chowdary, Bussa Sankeerth, Chennupati Kumar Chowdary, and Manu Gupta, delves into the realm of Machine Learning Operations (MLOps) to expedite the deployment of machine learning models. Deploying machine learning models can be a complex task, involving multiple factors such as continuous builds for efficiency and different libraries for predictions. As datasets grow to enhance predictive accuracy, certain parameters must be dynamically adjusted for performance tuning. Implementing these changes manually can be time-consuming and labor-intensive for developers. This study introduces an end-to-end automation cycle designed to streamline the deployment of machine learning models with enhanced performance.

**DESIGN THINKING**

Design Thinking Approach

Empathize:

Prior to addressing the problem, empathizing with end-users is paramount. Our primary users include cybersecurity experts and IT professionals. Understanding their priorities and how precise phishing detection can empower them is essential.

Actions:

- Conduct interviews and surveys with cybersecurity experts to grasp their needs and insights.

- Analyze historical phishing attack data to identify critical patterns and features.

- Collaborate with industry professionals to gain domain-specific knowledge.

Define:

Based on insights from the empathy phase, we will define clear objectives and success criteria for our project.

Objectives:

- Develop a phishing detection model with a false-positive rate of less than X%.

- Create a user-friendly web application for security professionals to input suspicious emails or URLs for immediate analysis.

Ideate:

Brainstorm creative solutions and techniques to tackle the problem. This phase involves exploring various machine learning algorithms and strategies for phishing detection.

Actions:

- Experiment with different machine learning algorithms, including decision trees, random forests, and deep learning models.

- Investigate feature selection methods to enhance model precision.

- Consider incorporating threat intelligence feeds for real-time phishing threat updates.

Prototype

Develop a prototype of the machine learning model and the user interface for phishing detection.

Actions:

- Build a Jupyter Notebook or Python script for data preprocessing, model training, and evaluation.

- Create a user-friendly web interface using tools like Flask or Django to allow users to submit email or URL data for analysis.

- Test the prototype with a subset of the dataset to verify it meets performance objectives.

Test :

Evaluate the model's performance using relevant metrics and gather feedback from users.

Actions:

- Split the dataset into training and testing sets.

- Train the model on the training set and evaluate it on the testing set.

- Utilize metrics like accuracy, precision, recall, and F1-score to assess model performance.

- Collect user feedback on the web interface for usability and effectiveness.

Implement

Once the prototype aligns with the defined objectives and garners positive feedback, proceed with full implementation.

Actions:

- Train the final machine learning model using the complete dataset.

- Deploy the model within a production-ready web application.

- Execute rigorous testing to ensure the application's resilience and ease of use.

Iterate

Continuously gather user feedback and iterate on the model and interface to improve accuracy and user experience.

Actions:

- Monitor the model's performance and update it as new phishing threats emerge.

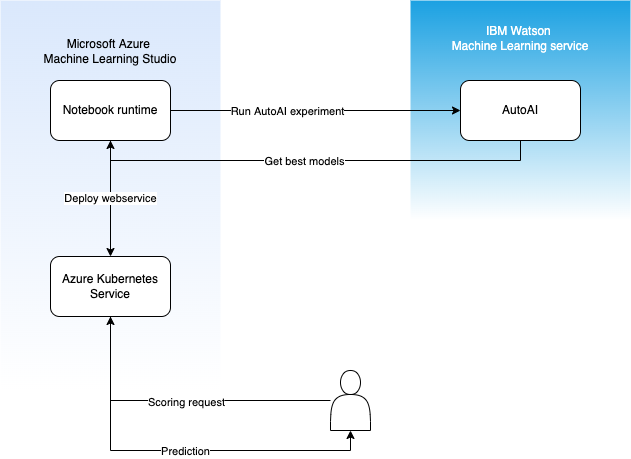
- Address user feedback promptly and make necessary enhancements to the web interface.

- Stay informed about advancementsin cybersecurity and phishing detection for potential improvements.

**TECHNOLOGY ARCHITECTURE**

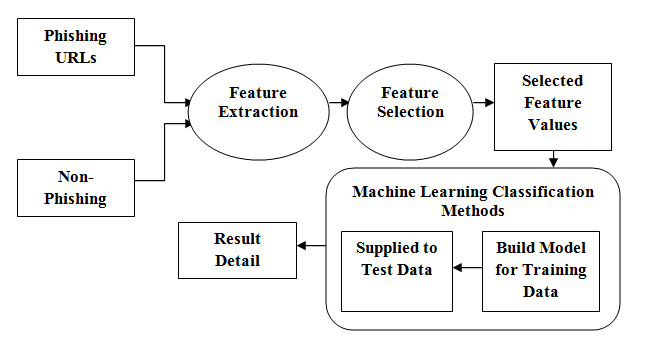
**A black and grey rectangular object with white text

Description automatically generated**



**Technology Architecture for Phishing Detection using IBM Cloud Watson Studio:**

The technology architecture for the phishing detection system using IBM Cloud Watson Studio encompasses various components and technologies working cohesively to provide a robust and efficient solution for identifying phishing websites and emails in real-time. Below is an overview of the architecture:



**1. Data Collection and Storage:**

- **Data Sources**: Utilize external sources, such as Kaggle, for obtaining datasets containing labeled phishing and benign examples.

- **Data Storage**: Store datasets in a cloud-based or on-premises database for easy access and retrieval.

**2. Data Preprocessing:**

- **Data Cleaning**: Clean the dataset by handling missing values, removing duplicates, and addressing data quality issues.

- **Feature Engineering**: Create relevant features from the dataset, such as extracting domain information, URL characteristics, and content analysis.

**- Data Transformation**: Perform encoding of categorical features, scaling of numerical features, and normalization as necessary.

**3. Model Development and Training:**

**- IBM Watson Studio**: Leverage IBM Watson Studio's AutoAI feature for model development, which includes:

- AutoML for model selection.

- Hyperparameter tuning.

- Model evaluation using various metrics like accuracy, precision, recall, and F1-score.

- Model versioning and management.

**4. Model Deployment:**

**- IBM Watson Studio Deployment Space:** Create a deployment space within IBM Watson Studio for deploying the selected machine learning model.

**- Web Service:** Deploy the model as a web service with an API endpoint for real-time predictions.

**5. Security and Access Control:**

**- API Authentication**: Implement API key-based authentication to secure access to the deployed model.

**- Access Control:** Configure permissions and access policies to control who can access the API and make predictions.

**6. Real-time Prediction:**

**- API Integration:** Develop a user-friendly interface or integrate the API with other systems and applications where real-time phishing detection is required.

**- Input Data:** Users or applications send data, such as URLs or email content, to the API for prediction.

**7. Monitoring and Maintenance:**

**- Performance Monitoring:** Continuously monitor the deployed model's performance in real-time.

**- Logging and Alerts:** Set up logging and alerts to detect anomalies or issues.

**- Model Updates:** Periodically update the model to adapt to evolving phishing techniques and threats.

**8. Scalability and Redundancy:**

**- Load Balancing**: Implement load balancing to distribute incoming requests evenly and ensure high availability.

**- Redundancy:** Deploy redundant instances of the model for failover and increased reliability.

**9. User Interface (Optional):**

- Develop a user interface for manual interactions, allowing users to input URLs or email content for phishing detection.

- The UI can provide user feedback and visualizations of detection results.

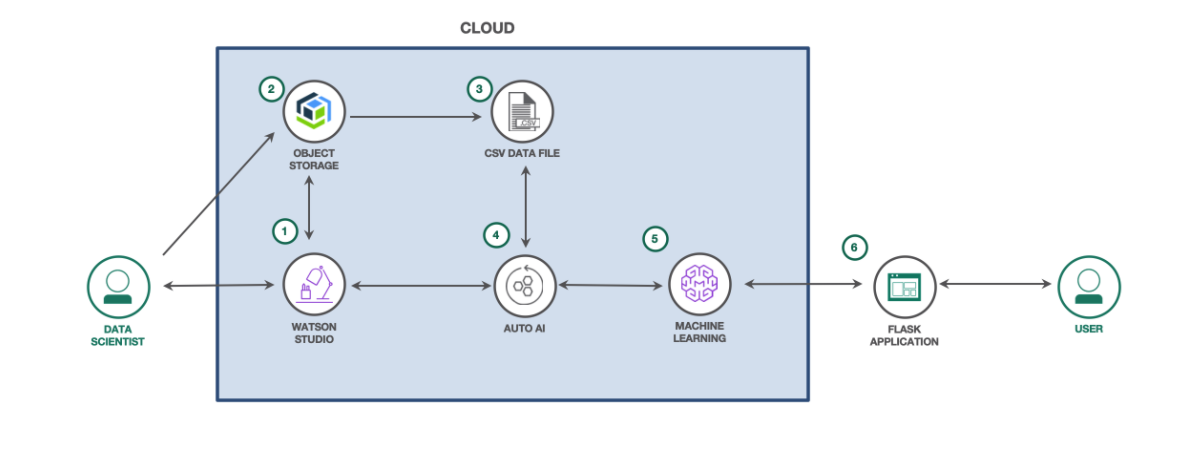
**10. Data Feedback Loop (Optional):**

- Collect feedback data on detected phishing attempts and false positives.

- Use this feedback to further train and improve the model.

**11. Reporting and Visualization (Optional):**

- Generate reports and visualizations to provide insights into the model's performance and effectiveness.



This technology architecture outlines a comprehensive solution for deploying a machine learning model for phishing detection. It focuses on scalability, security, real-time prediction, and ongoing maintenance to ensure the system's effectiveness in safeguarding online activities.

**MODULES DESCRIPTION**

**1. Data Collection and Storage Module:**

**- Objective:** This module focuses on gathering and storing datasets for training the phishing detection model.

**- Key Tasks:**

- Identify and access relevant data sources (e.g., Kaggle).

- Extract datasets containing labeled phishing and benign examples.

- Store datasets in a cloud-based or on-premises database.

**2. Data Preprocessing Module:**

**- Objective:** Prepare the dataset for model training by cleaning, transforming, and engineering features.

**- Key Tasks:**

- Data cleaning to handle missing values and remove duplicates.

- Feature engineering to create relevant features.

- Data transformation, including encoding categorical features and scaling numerical features.

**3. Model Development and Training Module:**

**- Objective:** Use IBM Watson Studio to develop and train machine learning models for phishing detection.

**- Key Tasks:**

- Utilize IBM Watson Studio's AutoAI feature for model selection.

- Perform hyperparameter tuning to optimize model performance.

- Evaluate model performance using metrics like accuracy, precision, recall, and F1-score.

**4. Model Deployment Module:**

**- Objective:** Deploy the selected machine learning model as a web service with an API endpoint.

**- Key Tasks:**

- Create a deployment space within IBM Watson Studio.

- Deploy the model to the cloud, making it accessible via the API.

- Configure the API endpoint for real-time predictions.

**5. Security and Access Control Module:**

**- Objective:** Ensure secure access to the deployed model.

**- Key Tasks:**

- Implement API key-based authentication to secure API access.

- Configure permissions and access policies for controlling user access.

**6. Real-time Prediction Module:**

**- Objective:** Enable users and applications to make real-time predictions using the deployed model.

**- Key Tasks:**

- Develop a user-friendly interface or integrate the API with other systems.

- Enable users to send input data (e.g., URLs or email content) for prediction.

**ALGORITHM AND TECHNOLOGY USED**

**1. Data Collection and Preprocessing:**

- Technology: Python (for data handling)

- Description: Collect the dataset from Kaggle, which includes features related to URLs and HTML content. Preprocess the data by handling missing values, encoding categorical features, scaling numerical features, and any other necessary data transformations.

**2. Model Development using AutoAI:**

- Technology: IBM Watson Studio (AutoAI)

- Algorithm: Automated Machine Learning (AutoML)

- Description: Leverage IBM Watson Studio's AutoAI feature to automate the model development process. AutoAI performs the following tasks:

- Feature engineering

- Model selection

- Hyperparameter tuning

- Cross-validation

**3. Model Evaluation and Selection:**

- Technology: IBM Watson Studio

- Algorithm: Various machine learning algorithms (selected by AutoAI)

- Description: AutoAI generates multiple candidate models using various machine learning algorithms (e.g., Random Forest, XGBoost, Logistic Regression). Evaluate these models using performance metrics like accuracy, precision, recall, and F1-score to select the best-performing model.

**4. Model Deployment:**

- Technology: IBM Watson Studio (AutoAI Deployment Space)

- Algorithm: None (deployment is not algorithm-based)

- Description: Create a deployment space in IBM Watson Studio, then deploy the selected machine learning model as a web service with an API endpoint for real-time predictions. This process involves packaging the model and its dependencies for deployment.

**5. Security and Access Control:**

- Technology: IBM Watson Studio

- Description: Implement security measures, including API key-based authentication, to secure access to the deployed model. Configure permissions and access policies to control who can use the API.

**6. Real-time Prediction:**

- Technology: HTTP/HTTPS, Python (for API integration)

- Description: Develop an interface or integrate the API with other systems or applications that require real-time phishing detection. Users or applications can send data (e.g., URLs or email content) to the API for prediction.

The use of AutoAI in IBM Watson Studio simplifies the machine learning model development process by automating various steps, including feature engineering, model selection, and hyperparameter tuning. While some steps involve traditional data processing and model evaluation, AutoAI streamlines the model development process. The deployment and operational aspects of the project focus on securing, maintaining, and scaling the model to provide real-time phishing detection.

**PROJECT DEVELOPMENT STEPS AND SCREENSHOT**

**Step 1: Account creation and create a new project in IBM Watson studio**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Step 2 : Choose the Dataset for to train the model**

**A screenshot of a computer

Description automatically generated**

**Step 3 :Train the Model**

**A screenshot of a computer

Description automatically generated**

**Step 4 : Choosing the Algorithm**

**A screenshot of a computer

Description automatically generated**

**Step 5 : ML Model Created Successfully in IBM Watson**

**A screenshot of a computer

Description automatically generated**

**Step 6: Create a new deployment space and Deploy the model as web service**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Step 7 :Test the Model using JSON Command**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Step 8 : Go to Deployment and get the API REFERENCE**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Step 9 :By using API Access the Model**

**A computer screen with many colorful text

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**API reference Link :** <https://private.us-south.ml.cloud.ibm.com/ml/v4/deployments/ba598ab2-eafe-4ec9-8c4b-90319209c5b3/predictions?version=2021-05-01>

**Api key :** edd3\_XlFXJaqKMIzn\_cPZhVcOlEhJ6S\_gZVL4CEExnz9

**Deployment Instructions:**

1. Create a Deployment Space: In your IBM Watson Studio environment, create a deployment space if you haven't already. This space will house the deployed model.

2. Deploy the Model: After creating a deployment space, deploy the machine learning model to that space. Ensure you have selected the model that you want to deploy.

3. Access the API Endpoint: Once the model is successfully deployed, you'll receive an API endpoint URL. This URL is the endpoint for making predictions using the deployed model.

**Using the API Endpoint for Predictions:**

You can use Python with the `requests` library to make API requests to the endpoint for predictions. Here's an example of how to do this:

```python

import requests

API\_KEY = 'YOUR\_API\_KEY'

# Request an access token

token\_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": API\_KEY, "grant\_type": 'urn:ibm:params:oauth:grant-type:apikey'})

mltoken = token\_response.json()["access\_token"]

# Define the API endpoint URL

api\_url = 'https://private.us-south.ml.cloud.ibm.com/ml/v4/deployments/ba598ab2-eafe-4ec9-8c4b-90319209c5b3/predictions?version=2021-05-01'

# Define the input data in the expected format

{

"input\_data": [

{

"fields": [

"id",

"NumDots",

"SubdomainLevel",

"PathLevel",

"UrlLength",

"NumDash",

"NumDashInHostname",

"AtSymbol",

"TildeSymbol",

"NumUnderscore",

"NumPercent",

"NumQueryComponents",

"NumAmpersand",

"NumHash",

"NumNumericChars",

"NoHttps",

"RandomString",

"IpAddress",

"DomainInSubdomains",

"DomainInPaths",

"HttpsInHostname",

"HostnameLength",

"PathLength",

"QueryLength",

"DoubleSlashInPath",

"NumSensitiveWords",

"EmbeddedBrandName",

"PctExtHyperlinks",

"PctExtResourceUrls",

"ExtFavicon",

"InsecureForms",

"RelativeFormAction",

"ExtFormAction",

"AbnormalFormAction",

"PctNullSelfRedirectHyperlinks",

"FrequentDomainNameMismatch",

"FakeLinkInStatusBar",

"RightClickDisabled",

"PopUpWindow",

"SubmitInfoToEmail",

"IframeOrFrame",

"MissingTitle",

"ImagesOnlyInForm",

"SubdomainLevelRT",

"UrlLengthRT",

"PctExtResourceUrlsRT",

"AbnormalExtFormActionR",

"ExtMetaScriptLinkRT",

"PctExtNullSelfRedirectHyperlinksRT"

],

"values": [

[1, 3, 2, 4, 45, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, "192.168.1.1", 1, 1, 1, 10, 20, 30, 40, 0, 5, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 10, 0, 0]

]

}

]

}

# Prepare headers with the access token

headers = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# Make a POST request to the API endpoint for predictions

response = requests.post(api\_url, json=input\_data, headers=headers)

# Print the prediction response

print("Scoring response:")

print(response.json())

```

**CONCLUSION**

In conclusion, the successful creation and deployment of a phishing detection machine learning model in IBM Watson Studio exemplify the transformative potential of technology in enhancing online security. Leveraging AutoAI, the model development process is streamlined, and with a dataset from Kaggle, the system is primed to swiftly recognize and mitigate evolving phishing threats. This proactive approach to safeguarding online interactions not only empowers users and organizations but also underscores the value of leveraging advanced machine learning solutions in an era where cybersecurity is paramount.