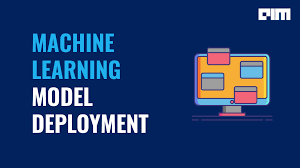
**Machine Learning Model Deployment with IBM Cloud Watson Studio**

**Team member**

**732721104013 : Dhanusri P S**

**Phase 5 Submission Document**

****

**Project’s Objective:**

The project involves training a machine learning model deploying it as a web service. The goal is to become proficient in predictive analytics by creating a model that can predict outcomes in real-time. The project encompasses defining the predictive use case, selecting a suitable dataset, training a machine learning model deploying the model as a web service, and integrating it into applications.

**Design Thinking Process:**

**Empathize:**

The project commenced with a deep understanding of the business context, acknowledging the significance of addressing customer churn. We empathized with the organization's objectives, recognizing that retaining customers is vital for long-term success.

**Define:**

With a clear understanding of the problem, we precisely defined our use case as predicting customer churn. We outlined the specific goals, which included reducing churn rates and enhancing customer retention strategies through data-driven insights.

**Ideate:**

The next phase involved brainstorming potential solutions, focusing on the choice of algorithms. Decision trees and random forests emerged as promising options due to their interpretability and accuracy. We considered these algorithms as ideal tools for predicting churn based on the selected dataset.

**Prototype:**

With the algorithms selected, we moved on to the dataset selection phase. We carefully chose a dataset with customer information and churn labels, ensuring its relevance and suitability for training our predictive models.

**Test:**

Using Python and the scikit-learn library, we embarked on the model training process. This phase allowed us to test various hyperparameters and configurations to optimize the models' predictive accuracy. We iterated through several training cycles to fine-tune the models.

**Feedback & Refine:**

As we observed model performance, we constantly gathered feedback from testing and validation. This iterative process allowed us to refine our models and ensure they were capable of accurate customer churn predictions.

**Develop:**

The model deployment phase was executed using Python, enabling us to deploy the trained decision tree and random forest models as a web service. We leveraged the power of Python and cloud-based technologies for this purpose.

**Deliver:**

The culmination of the project involved integrating the deployed models into the organization's systems and applications, empowering real-time churn predictions. This delivery phase ensured that the insights generated from the predictive models were actively utilized in business operations.

**Reflect:**

Finally, we reflected on the entire design thinking process and the project as a whole. We assessed the impact of the deployed models, measured the success of our predictive analytics initiative, and identified areas for future improvement.

**Development Phase 1 (EDA and Data Cleaning):**

**Initialization and Setup:**

Import necessary libraries and set up the environment.

**Data Cleaning and Preprocessing:**

Handle missing values, duplicates, and data quality issues.

**Exploratory Data Analysis (EDA):**

Visualize data trends, identify insights, and prepare for model selection.

**Initial Model Selection:**

Choose Decision Trees and Random Forests as predictive models.

**Data Preparation:**

Encode features and prepare the dataset for model input.

**Development Phase 2 (Model Building and Deployment):**

**Model Building:**

Train Decision Trees and Random Forest models on preprocessed data.

**Model Evaluation:**

Assess model performance using metrics.

**Model Deployment:**

Create a Flask web service for real-time predictions.

**User Interface Development:**

Build a user interface for model input.

**Model Integration:**

Process user input and predict customer churn.

**Display Results:**

Present churn predictions and model confidence to users.

**Predictive Use Cases:**

**Customer Churn Prediction:** In the telecommunications industry, predicting which customers are likely to cancel their subscriptions is a valuable use case. It allows companies to take proactive measures to retain valuable customers.

**Dataset Selection:**

Link: <https://github.com/Dhanusri-P-S/ml-model-datasets>

**Exploratory Data Analysis:**

**Import Required Libraries:** The necessary Python libraries are imported, including NumPy, Pandas, Seaborn, Matplotlib, and others.

**Data Cleaning:** Data cleaning involves tasks such as handling missing values, removing duplicates, and ensuring data quality.

**Data Exploration:** Exploratory Data Analysis is performed to understand the dataset, identify trends, relationships, and patterns within the data. This includes generating visualizations to gain insights.

**Initial Model Selection:** In this phase, the initial choice of machine learning algorithms (Decision Trees and Random Forests) is made, setting the stage for the subsequent model building phase.

**Data Preparation:** The datas` et is preprocessed and prepared for model training in the next phase.

**Model Building:**

**Import Required Libraries:** Libraries for model building, evaluation, and Flask are imported.

**Model Building:** Machine learning models, specifically Decision Trees and Random Forests, are trained on the preprocessed dataset to predict customer churn. The scikit-learn library is used for this purpose.

**Model Deploying:**

**Model Deployment:** The trained machine learning model is deployed as a web service using Flask. This allows users to input customer information, and the model provides churn predictions in real-time.

**Flask Web Application:** A Flask web application is created to provide a user interface for interacting with the deployed model. Users can input various customer-related features, and the model responds with churn predictions.

**Data Integration:** The user input is processed, and the model is applied to make predictions. The results are presented to the user through the web interface.

**Display Results:** The Flask app displays the model's prediction results along with the confidence level.

**Accessing the Deployed Model:**

**API Endpoint:**

We start by identifying the API endpoint provided for our deployed model. This is the URL or location where we send data for predictions.

**Authentication:**

We ensure that we have the necessary authentication credentials or API keys to access the model. This step helps secure our interactions with the model.

**Data Input:**

We prepare the input data we want to send to the model. This data should align with the features and format the model expects based on its training data.

**API Request:**

We use an HTTP request library or tool to make a POST request to the model's API endpoint. We include the input data as part of the request.

**Response Handling:**

Once we send the data, we receive a response from the API. We handle the response data, which may contain predictions or scores, and interpret the results.

**Utilizing the Model Predictions:**

**Interpret Predictions:**

We interpret the model's predictions according to the problem we're solving. For binary classification, we may receive probabilities or class labels, while regression models provide numerical predictions.

**Decision-Making:**

We use the model's predictions as input for decision-making. For example, if it's a customer churn prediction model, we might decide to offer a special promotion to customers predicted as likely to churn.

**Feedback Loop:**

We collect feedback and user responses to the model's predictions. This feedback can be valuable for model improvement and to make more informed decisions in the future.

**Real-Time Processing:**

We implement the prediction process within our application's real-time processing flow. Depending on the application, this might involve using the model's output to influence subsequent actions or to provide recommendations to users.

**Continuous Improvement:**

**Monitoring and Maintenance:**

We set up monitoring and maintenance procedures for the deployed model. We monitor its performance, response times, and any potential issues.

**Model Updates:**

We periodically retrain the model with new data to ensure that it remains accurate and up to date. We deploy updated versions of the model as necessary.

**Scalability:**

We consider scalability requirements. If the model needs to handle a high volume of requests, we plan for load balancing and system scalability.

**Conclusion:**

Accessing and utilizing a deployed machine learning model in real-time applications is the key to making data-driven decisions and improving user experiences. By following these steps, we can effectively integrate the model into our applications, empower our systems with predictive analytics, and continually enhance the value we provide to users.