PROJECT TITLE: **REAL-TIME PREDICTIVE ANALYTICS WITH IBM CLOUD WATSON STUDIO**

NAME OF PROJECT: CUSTOMER CHURN ANALYSIS

PROJECT OBJECTIVE:

The objective of a customer churn analysis project is to understand and reduce the rate at which customers stop using a product or service. Churn, also known as customer attrition, is a critical metric for businesses, as it directly impacts their revenue and profitability. The primary goals of a customer churn analysis project typically include:

1. Identify Churn Patterns: Analyze historical data to identify patterns and trends that lead to customer churn. This can involve examining customer behavior, demographics, usage patterns, and transaction history.
2. Predict Churn: Develop predictive models that can forecast which customers are at a high risk of churning in the near future. Machine learning algorithms, such as logistic regression, decision trees, or neural networks, can be employed for this purpose.
3. Segment Customers: Divide the customer base into segments based on various criteria, such as demographics, customer tenure, usage patterns, or behavior. Understanding which customer segments are more likely to churn can help in targeted retention efforts.
4. Root Cause Analysis: Investigate the reasons behind churn by analyzing customer feedback, surveys, and support tickets. Identifying the key drivers of churn can guide strategies to mitigate it.
5. Customer Engagement: Develop strategies to engage and retain customers proactively. This might involve personalized marketing, loyalty programs, improved customer support, or product enhancements.
6. Retention Strategies: Design and implement specific retention strategies to reduce churn, such as discounts, upselling, cross-selling, or tailored communications to at-risk customers.
7. Monitoring and Evaluation: Continuously monitor and evaluate the effectiveness of churn reduction strategies. Adjust and refine these strategies based on ongoing data analysis and feedback.
8. Measurement of Success: Define clear metrics to measure the success of the churn reduction efforts, such as a decrease in churn rate, increased customer lifetime value, or improved customer satisfaction.
9. Data-Driven Decision Making: Ensure that the project is data-driven, with decisions and strategies based on analytical insights and evidence.
10. Reporting and Visualization: Create reports and dashboards to communicate the findings and progress of the churn analysis to relevant stakeholders within the organization.

DESIGN AND THINKING PROCESS:

Designing and conducting a customer churn analysis involves a structured thinking process and several key steps to ensure a successful project. Here's a step-by-step guide to help you design and execute a customer churn analysis:

1. Define Objectives:
    - Start by clearly defining the objectives of the churn analysis. What do you want to achieve with this project? Is it to reduce churn, improve customer retention, or better understand the reasons behind customer attrition?
2. Data Collection:
    - Gather relevant data sources. This may include customer demographics, transaction history, usage data, customer support interactions, feedback surveys, and any other information that can provide insights into customer behavior and churn.
3. Data Preprocessing:
    - Clean and preprocess the data to ensure its quality and consistency. This may involve handling missing values, removing outliers, and transforming data into a suitable format for analysis.
4. Feature Selection:
    - Identify the relevant features or variables that might impact churn. This could include customer demographics, purchase frequency, customer lifetime value, satisfaction scores, and more.
5. Data Exploration:
    - Perform exploratory data analysis (EDA) to understand the data better. Visualize data distributions, correlations, and patterns that might be related to churn. EDA can help uncover initial insights and hypotheses.
6. Predictive Modeling:
    - Develop predictive models to forecast customer churn. Common machine learning algorithms like logistic regression, decision trees, random forests, or neural networks can be used. Train and evaluate these models using historical data.
7. Model Evaluation:
    - Assess the model's performance using appropriate metrics, such as accuracy, precision, recall, F1-score, or the area under the ROC curve. Choose the evaluation metrics that align with your project's goals.

DEVELOPMENT PHASES:

PHASE 1: DEFINING THE PREDICTIVE USE CASE OF THE PROJET

PHASE 2: THE INNOVATION PHASE OF OUR PROJECT WHICH WILL DEFINE THE PLATFORM AND DATASET SELECTION

PHASE 3&4: DEVELOPMENT PHASES IN WHICH INCLUDES THE DATASET CLEANING,DEFINING OUTLIERS, VISUALISING THE DATASET, AND THE TRAINING OF THE MODEL

PREDICTIVE USE CASE:

Predictive use cases of customer churn analysis involve forecasting which customers are at risk of leaving a product or service in the future. By identifying and predicting potential churners, businesses can take proactive steps to retain these customers and mitigate revenue loss

- Customer Segmentation:
    - Predictive analysis can help segment customers based on their churn risk. By categorizing customers into high-risk, medium-risk, and low-risk groups, businesses can tailor their retention strategies and communications accordingly. High-risk customers may receive more aggressive retention efforts, while low-risk customers may receive more personalized offers.
- Personalized Retention Efforts:
    - Predictive models enable businesses to craft personalized retention strategies for individual customers. These strategies may include offering discounts, incentives, loyalty programs, or tailored product recommendations to address specific reasons for potential churn.
- Product and Service Improvements:
    - Churn prediction can also highlight specific pain points or issues that are causing customers to leave. This information can guide product and service improvements to address these concerns and enhance the overall customer experience.

DATASET SELECTION:

Selecting the right dataset for customer churn analysis is a critical step in ensuring the success and accuracy of your analysis and predictive modeling

Dataset

https://www.kaggle.com/becksddf/churn-in-telecoms-dataset/data

in kaggle there are different dataset for our project,we can select any of the dataset which fit our model

we have selected the dataset from the github repository

MODEL TRAINING:

Analyze and preprocess the data

Watsonx provides the tool for the preprocessing of the data to improve the quality of the dataset and to get the appropriate output so it is necessary to analyze and preprocess the data

We use the SPSS modeler to create our project,this modeler enable you to quickly develop predictive models using business expertise and deploy them into business operations to improve the decision making , the Watson studio provides this modeler and helps to create our project To move to the further processing of our model we need to move towards the modeler flow,it is a tool provided by the Watson to help with the mode

DEPLOYMENT PROCESS:

The deployment process of a customer churn analysis model involves making the predictive model available for real-time or batch predictions within your organization's operational systems. Here are the steps for deploying a customer churn analysis model:

1. **Select Deployment Environment**:
    - Choose where you want to deploy your model. Common options include cloud-based platforms, on-premises servers, or edge devices, depending on your organization's infrastructure and needs.
2. **Model Export**:
    - Export the trained customer churn prediction model from your development environment (e.g., your data science platform like IBM Watson Studio) in a format that can be used in the deployment environment. Common formats include PMML (Predictive Model Markup Language), ONNX (Open Neural Network Exchange), or a serialized model object in a programming language like Python.

- **Export the Churn Model**:

    - Ensure you have a trained customer churn prediction model that's compatible with IBM Watson. Export the model in a suitable format, such as PMML, ONNX, or a serialized model object.

- **Set Up IBM Watson Services**:

    - Make sure you have access to IBM Watson services, such as IBM Watson Machine Learning and IBM Watson Studio. These services provide the infrastructure and tools needed for model deployment and integration.

- **Create an API for Real-Time Predictions**:

- If you want to make real-time predictions, you need to create an API endpoint for your model. IBM Watson Machine Learning allows you to deploy models as web services, which can be accessed via REST APIs.

a. In IBM Watson Studio, navigate to your deployment space and create a deployment for the churn model. b. Configure the deployment settings, specifying that it's for online (real-time) predictions. c. Deploy the model to create an API endpoint.

- **Batch Scoring Configuration**:

  - If you need to perform batch predictions (e.g., for a large customer dataset), configure the batch scoring process using IBM Watson services. You can schedule and run batch predictions on your data.

a. In IBM Watson Studio, navigate to your deployment space and create a batch deployment for the churn model. b. Configure the deployment settings, specifying the batch scoring mode. c. Upload the batch input data and run the batch scoring job.

- **API Key and Authentication**:

  - Secure your API endpoints by setting up authentication and authorization mechanisms. IBM Watson provides options to secure your API endpoints, including API keys, OAuth, or other authentication methods.

- **Integration with Applications**:

  - Integrate the API endpoints into your existing applications, customer relationship management (CRM) systems, or other relevant platforms.

a. Develop or modify your applications to include API calls to the churn model for real-time predictions. You will need to use the API key and the API endpoint URL provided by IBM Watson.

b. If you're using batch scoring, create a process to automatically send batch data to the IBM Watson API for batch predictions. This could be done using scheduled jobs or scripts.

SHORT DESCRIPTION OF REALTIME USE OF MODEL:

1. **Prediction Output**:
   - The API receives the prediction from the model and returns it to the requesting application or system. The output may include the churn prediction itself (e.g., "churn" or "not churn") along with a confidence score or probability.
2. **Utilization in Applications**:
   - The application or system that sent the request can then utilize the churn prediction for various purposes, such as customizing user experiences, triggering retention strategies, or making real-time decisions. For example:

    - If the model predicts that a customer is at high risk of churning, the application might trigger a pop-up message with a special offer to retain the customer.
    - In a customer service application, the model's predictions can prioritize high-risk customers for personalized support.
    - In a marketing application, the model's predictions can guide targeted marketing campaigns and promotions.
3. **Logging and Monitoring**:
    - To maintain quality and track the performance of the real-time predictions, logging and monitoring are essential. Logs are generated to record incoming requests, data used for predictions, and prediction results. Monitoring tools can help identify issues, anomalies, and changes in prediction accuracy.
4. **Feedback Loop**:
    - The real-time predictions and user interactions with the model provide valuable data for feedback and model improvement. Continuous feedback is used to retrain the model and enhance its accuracy over time.

Accessing and utilizing a deployed customer churn analysis model for real-time predictions enables organizations to take proactive measures to retain customers, reduce churn, and optimize customer experiences based on up-to-the-minute insights.

GitHub repository link: https://github.com/imshaaa/CAD101.git

DIRECT LINLK OF MODEL: https://private.eu-de.ml.cloud.ibm.com/ml/v4/deployments/customer/predictions?version=2021-05-01

DEPLOYMENT STEPS:

Deployment Steps:

1. **Train and Save Your Machine Learning Model**:
    - First, make sure you have a trained machine learning model for customer churn analysis. You can train this model using tools like scikit-learn. Once trained, save the model to a file, such as a pickle file (.pkl) or any other preferred format.
2. **Set Up a Web Service Framework**:
    - You'll need a web framework to create the API that exposes your model as a service. Flask is a lightweight and popular choice. If you haven't already, install Flask using `pip`:

```
bash
pip install flask
```

- **Create a Python Script for the Web Service**:
    - Write a Python script that defines your web service using Flask. This script should include endpoints for receiving input data and returning predictions from your model.

Here's a basic example of a Flask web service script:

```
python
```
- `from flask import Flask, request, jsonify`

```
import joblib  # Use joblib to load your model

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.get_json()  # Get input data from the request
        model = joblib.load('your_model.pkl')  # Load your trained model
        prediction = model.predict(data)
        return jsonify({'prediction': prediction.tolist()})
    except Exception as e:
        return jsonify({'error': str(e)})

if __name__ == '__main__':
    app.run(debug=True)
```

Modify this script to match your specific model and data requirements.

- **Run the Web Service**:
    - Start the web service by running the script you created. This will launch a local web server that listens for incoming requests.

```
bash
python your_web_service_script.py
```

To make API requests for making predictions in a customer churn analysis, you'll typically need to send a POST request to the API endpoint you've set up. Here's an example of how to structure API requests using Python's `requests` library to obtain churn predictions:

Assuming you have a Flask web service deployed as shown in the previous example, here's how to make API requests:

```
python
import requests
import json

# Input data for prediction (modify to match your dataset's features)
input_data = {
    "feature1": 42,
    "feature2": 0.85,
    "feature3": "High",
    "feature4": "Yes"
}

# Define the API endpoint URL (replace with your actual endpoint)
api_url = "http://localhost:5000/predict"

# Send a POST request with the input data to the API
response = requests.post(api_url, json=input_data)
```

```
# Check the response status code
if response.status_code == 200:
    # Parse the JSON response to obtain the prediction
    result = response.json()
    churn_prediction = result['prediction']
    print(f"Churn Prediction: {churn_prediction}")
else:
    print(f"API Request Failed. Error: {response.text}")
```

In this example:

- `input_data` is a dictionary containing the input features for a customer's churn prediction. Modify this dictionary to match the features used by your model.
- `api_url` should be set to the actual URL of your API endpoint. Replace `"http://localhost:5000/predict"` with the correct URL.
- A POST request is sent to the API endpoint with the `input_data` provided as JSON in the request body.
- The response status code is checked. If the request is successful (status code 200), the prediction is extracted from the JSON response and printed.

Please note that in a real-world scenario, you would replace the example input features with actual data from your customer churn analysis dataset, and you would use the correct API endpoint URL to access your deployed model. Additionally, handle any errors or exceptions that may occur during the API request process.