MACHINE LEARNING MODEL DEPLOYMENT WITH IBM CLOUD WATSON STUDIO

Predictive Use Case: Customer Churn Prediction

Development part 1:

Dataset:

For this example, let's use a fictional CSV dataset containing customer information such as age, monthly spend, usage patterns, and churn status (1 for churned, 0 for not churned).

Step 1: Import the Dataset in Watson Studio

- ***Open Watson Studio:***
  Go to IBM Cloud and open your Watson Studio project.

- ***Create a New Notebook:***
  Inside your project, create a new Jupyter Notebook.

- ***Import Dataset:***
  Import the dataset into your notebook using Pandas.

python

```
import pandas as pd
df = pd.read_csv("path/to/your/dataset.csv")
```

Step 2: Data Preprocessing and Feature Selection

Data Cleaning:

- Handle missing values if any.
- Convert categorical variables into numerical representations if needed (using techniques like one-hot encoding).

Feature Selection:

- Identify relevant features for prediction. For example:

python

```
features = ['Age', 'MonthlySpend', 'UsagePattern']
```

```
X = df[features]
y = df['Churn']
```

Step 3: Model Training

- ***Split Data:***
  - Split the data into training and testing sets.

python

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data, Y_data, test_size=0.3,
random_state=0)
```

Output:

```
             gender           category
SeniorCitizen      category
Partner            category
Dependents         category
tenure             int64
MultipleLines      category
InternetService    category
OnlineSecurity     category
OnlineBackup       category
DeviceProtection   category
TechSupport        category
StreamingTV        category
StreamingMovies    category
Contract           category
PaperlessBilling   category
PaymentMethod      category
MonthlyCharges     float64
TotalCharges       float64
```

- ***Choose and Train a Model:***
    - Choose a machine learning algorithm and train the model.

Python

```
logreg = LogisticRegression(max_iter=300)
logreg.fit(X_train, Y_train.values.ravel())
```

output:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=300, multi_class='warn',
n_jobs=None, penalty='l2', random_state=None, solver='warn',
tol=0.0001, verbose=0, warm_start=False)
```

Step 4: Model Evaluation and Fine-Tuning

- ***Evaluate Model:***
    - Evaluate the model's performance on the test data.

python

```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set:
{:.2f}'.format(logreg.score(X_test, Y_test)))
print(classification_report(y_test, y_pred))
```

Output:

Accuracy of logistic regression classifier on test set: 0.77

	precision	recall	f1-score	support
0	0.83	0.75	0.79	1064
1	0.78	0.85	0.81	1101
micro avg	0.80	0.80	0.80	2165
macro avg	0.80	0.80	0.80	2165
weighted avg	0.80	0.80	0.80	2165

- ***Fine-Tuning:***
    - If necessary, fine-tune the model parameters for better performance.

Step 5: Model Deployment

- ***Create a Deployment Space:***
  - Create a deployment space within your Watson Studio project.

- ***Deploy Model:***
  - Deploy the trained model to the deployment space.

python

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient

wml_credentials={

   "apikey": "*****************************",

   "instance_id": "************************",

   "url": "********************************"

}

client = WatsonMachineLearningAPIClient(wml_credentials)

model_props={

   client.repository.ModelMetaNames.NAME: "Logistic Regression Churn model",

   client.repository.ModelMetaNames.AUTHOR_EMAIL: "diegoramirez@gmail.com",

   client.repository.ModelMetaNames.FRAMEWORK_VERSION: "0.20",

   client.repository.ModelMetaNames.FRAMEWORK_NAME: "scikit-learn"

}


model_artifact=client.repository.store_model(logreg, meta_props=model_props)

client.repository.list()
```

Output:

```
------------------------------------  ------------------------------  ------------------------  -----------------
-----
GUID                          NAME                       CREATED                FRAMEWORK
TYPE
f1cf615d-d9a9-436c-9771-88df97c7e6ec  Logistic Regression Churn model  2020-05-
20T02:43:02.470Z  scikit-learn-0.20  model
------------------------------------  ------------------------------  ------------------------  -----------------
-----
```

Step 6: Test the Deployed Model

- ***Get Deployment Endpoint:***
  - Retrieve the endpoint URL for the deployed model.

python

```
#Get model UID
published_model_uid = client.repository.get_model_uid(model_artifact)
#Deploy the model
created_deployment = client.deployments.create(published_model_uid,
name="ChurnModelDeployment")
```

Output:

```
##########################################################################
###################

Synchronous deployment creation for uid: 'f1cf615d-d9a9-436c-9771-
88df97c7e6ec' started

##########################################################################
###################


INITIALIZING
DEPLOY_SUCCESS


-------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='f9e80285-841e-4783-
bea0-0c76bf8a8ec4'
-------------------------------------------------------------------------------------------
```

- ***Test the API Endpoint:***
    - Use the endpoint URL to make predictions.

python

```
scoring_payload = {"fields": list(X_test.columns),
   "values":X_test.iloc[11:20].values.tolist()}
predictions = client.deployments.score(scoring_endpoint, scoring_payload)
print(predictions)
```

Output:

{'fields': ['prediction', 'probability'], 'values': [[0, [0.9849121857890184, 0.01508781421098155]], [0, [0.7668201230777614, 0.23317987692223857]], [0, [0.9977147998805967, 0.0022852001194032597]], [0, [0.975127668806959, 0.024872331193040997]], [0, [0.7327641504178833, 0.26723584958211666]], [0, [0.9916415671999173, 0.008358432800082749]], [1, [0.37651074677061636, 0.6234892532293836]], [0, [0.9986890733149208, 0.0013109266850791436]], [0, [0.9828675236249786, 0.017132476375021317]]]}

Conclusion:

This completes the process of deploying a customer churn prediction model using IBM Cloud Watson Studio's tools. Ensure that you replace placeholders like `"your-api-key"` and `"your-instance-id"` with your actual IBM Cloud credentials and follow the correct syntax for your specific dataset and model requirements.