# Integrating the Predictive Model with Tableau using TabPy

This guide outlines the steps to connect Tableau to a running TabPy server and use the Python function for predictions.

## Prerequisites

Before you begin, ensure that you have **Python 3.6 or newer** installed on your local machine. This is a crucial requirement for the TabPy library.

To check your Python version, open your command prompt or terminal and run the following command:

python --version

## Step 1: Install the TabPy Library

Before you can start the server, you need to install the TabPy library.

1. Open your command prompt or terminal.
2. Run the following command to install the library:  
   pip install tabpy

## Step 2: Start the TabPy Server

Before you do anything in Tableau, ensure your TabPy server is running.

1. Open your command prompt or terminal.
2. Run the following command:  
   tabpy

## Step 3: Run the Python notebook to train & save the prediction model.

## 

## Step 4: Run the Python notebook to load & deploy the prediction function to the TabPy server

This is a critical step that you must perform after starting the TabPy server. This script loads your trained model and deploys the prediction function to the TabPy server, making it available for Tableau to call.

1. Start the Jupyter notebook.
2. Execute the capstone\_patient\_readmission\_tabpy\_deploy\_script file.

## Step 5: Configure the Analytics Extension in Tableau

1. Open Tableau Desktop.
2. Go to the menu and navigate to **Help > Settings and Performance > Manage Analytics Extension Connection**.
3. In the dialog box, select **TabPy / External API**.
4. For **Server**, enter localhost.
5. For **Port**, enter 9004.
6. Click **Test Connection**. If the test is successful, a confirmation message will appear. Click **OK**.

## 

## Step 6: Create Parameters for User Input

To allow a user to interact with the prediction and see how changes in a patient's characteristics affect their readmission risk, you can use **Parameters**. This is a great way to show the power of your model without requiring a user to edit the data source.

First, create a separate parameter for each of the features you want the user to control.

1. In the Tableau Data pane, right-click and select **Create Parameter...**.
2. Give the parameter a clear name, such as Enter Age.
3. Set the **Data type** to Integer and **Allowable values** to All.
4. Repeat this process for other features, like Enter Number of Visits and Enter Medication Count.
5. To make these visible on your dashboard, right-click each new parameter and select **Show Parameter Control**.

### Creating a Drop-Down for Categorical Features

To create a drop-down list for your binary and one-hot encoded features, you'll follow a similar process, but you will set the data type and allowable values differently.

#### 1. Binary Features (yes/no or true/false)

For features like diabetes\_med\_yes, has\_procedure\_performed, etc., you'll create a parameter that gives the user two choices.

* Right-click in the Data pane and select **Create Parameter...**.
* Name it something like Has Diabetes Medication.
* Set the **Data type** to Boolean.
* Under **Allowable values**, choose List.
* Click the **Add from Field** button and find a boolean field from your dataset, or manually type in True and False as your values.
* The parameter control will now appear as a simple drop-down list.

#### 2. Categorical Features (like Diagnosis)

For features that are part of a one-hot encoded group, such as diag\_1, you can create a single parameter with a list of all the possible values.

* Right-click and select **Create Parameter...**.
* Name it Select Primary Diagnosis.
* Set the **Data type** to String.
* Under **Allowable values**, choose List.
* In the list, you can manually add the names of the categories: Diabetes, Digestive, Injury, Musculoskeletal, Other, Respiratory.

#### 3. The Age Feature

For the age feature, you have a choice.

* **Option A: Drop-down List (Categorical)**: If you want to present the user with a drop-down list of the defined age groups (e.g., 40-50, 50-60), create a parameter with a String data type and a List of these values.

**4. The Parameters in Tableau**

I’ve created the following parameters in a Tableau workbook:

1. [Select Age Group], (List)

2. [Has Inpatient Visit], (Boolean)

3. [Has Outpatient Visit], (Boolean)

4. [Has Emergency Visit], (Boolean)

5. [Has Procedure Performed], (Boolean)

6. [Prescribed Diabetes Med], (Boolean)

7. [Select Medical Specialty], (List)

8. [Select Primary Diagnosis], (List)

9. [Select Secondary Diagnosis], (List)

10. [Select Tertiary Diagnosis], (List)

## 

## Step 7: Create a Calculated Field in Tableau

Now that Tableau is connected, you can call your Python function from a calculated field using the parameters you just created.

1. In your Tableau workbook, navigate to the worksheet you want to use the prediction on.
2. Right-click on the data pane and select **Create Calculated Field...**.
3. Name the calculated field something descriptive, like Predicted Readmission Risk.
4. Enter the following simplified formula:
   * The SCRIPT\_REAL command tells Tableau to expect a floating-point number (a decimal) back from Python.
   * The next arguments are the patient features you are sending to the function. These should now be your newly created parameters.

SCRIPT\_REAL(

"

import pandas as pd

import numpy as np

# Get the selected values from the Tableau parameters and fields

# Remember that each \_argN is a list from Tableau, so access the first element [0]

try:

selected\_age\_group = \_arg1[0]

has\_inpatient\_visit = \_arg2[0] # Boolean

has\_outpatient\_visit = \_arg3[0] # Boolean

has\_emergency\_visit = \_arg4[0] # Boolean

has\_procedure\_performed = \_arg5[0] # Boolean

prescribed\_diabetes\_med = \_arg6[0] # Boolean

selected\_medical\_specialty = \_arg7[0] # String from List parameter

selected\_diag\_1 = \_arg8[0] [0] # String from List parameter - Added [0] here as well, in case it's nested

selected\_diag\_2 = \_arg9[0] [0] # String from List parameter - Added [0] here as well

selected\_diag\_3 = \_arg10[0] [0] # String from List parameter - Added [0] here as well

# Assuming \_arg11 onwards are other necessary feature fields from your data source

# Print input values for debugging

print(f'Input received: age\_group={selected\_age\_group}, inpatient={has\_inpatient\_visit}, outpatient={has\_outpatient\_visit}, emergency={has\_emergency\_visit}, procedure={has\_procedure\_performed}, diabetes\_med={prescribed\_diabetes\_med}, medical\_specialty={selected\_medical\_specialty}, diag\_1={selected\_diag\_1}, diag\_2={selected\_diag\_2}, diag\_3={selected\_diag\_3}')

except Exception as e:

print(f'Error receiving input data from Tableau: {e}')

# Return a list of None or a default value to indicate an error

return [None] \* len(\_arg1) # Assuming \_arg1 is always present

# Create a dictionary to hold the data for the DataFrame

# Initialize with default values (0 for one-hot encoded, False for boolean)

data = {

'age\_[50-60)': 0

,'age\_[60-70)': 0

, 'age\_[70-80)': 0

, 'age\_[80-90)': 0

, 'age\_[90-100)': 0,

'has\_inpatient\_visit': has\_inpatient\_visit,

'has\_outpatient\_visit': has\_outpatient\_visit,

'has\_emergency\_visit': has\_emergency\_visit,

'has\_procedure\_performed': has\_procedure\_performed,

'diabetes\_med\_yes': prescribed\_diabetes\_med, # Mapping parameter name to feature name

# Initialize one-hot encoded columns for medical\_specialty, diag\_1, diag\_2, diag\_3 to 0

# You'll need to list all possible values from your training data here

'medical\_specialty\_Emergency/Trauma': 0,

'medical\_specialty\_Family/GeneralPractice': 0,

'medical\_specialty\_InternalMedicine': 0,

'medical\_specialty\_Missing': 0, # Include if you kept 'Missing' in your model

'medical\_specialty\_Other': 0,

'medical\_specialty\_Surgery': 0,

# ... add all other medical\_specialty one-hot encoded columns

'diag\_1\_Diabetes': 0, 'diag\_1\_Digestive': 0, 'diag\_1\_Injury': 0

, 'diag\_1\_Musculoskeletal': 0, 'diag\_1\_Other': 0, 'diag\_1\_Respiratory': 0,

'diag\_2\_Diabetes': 0, 'diag\_2\_Digestive': 0, 'diag\_2\_Injury': 0,

'diag\_2\_Musculoskeletal': 0, 'diag\_2\_Other': 0, 'diag\_2\_Respiratory': 0,

'diag\_3\_Diabetes': 0, 'diag\_3\_Digestive': 0, 'diag\_3\_Injury': 0,

'diag\_3\_Musculoskeletal': 0, 'diag\_3\_Other': 0, 'diag\_3\_Respiratory': 0,

# ... add all other diag\_3 one-hot encoded columns

# ... include any other feature columns from your Tableau data source if needed

# e.g., 'time\_in\_hospital': \_arg11 # Assuming time\_in\_hospital is \_arg11

}

# Perform one-hot encoding based on the selected parameter values

if selected\_age\_group in data:

data[selected\_age\_group] = 1

# For medical\_specialty, diag\_1, diag\_2, diag\_3, you need to map the selected string value

# from the parameter to the corresponding one-hot encoded column name.

# This requires knowing how the strings in your Tableau List parameters

# map to the column names created by pd.get\_dummies.

# Example mapping (adjust based on your actual column names):

medical\_specialty\_col = 'medical\_specialty\_' + selected\_medical\_specialty.replace(' ', '').replace('/', '') # Example: 'Emergency/Trauma' -> 'medical\_specialty\_EmergencyTrauma'

if medical\_specialty\_col in data:

data[medical\_specialty\_col] = 1

diag\_1\_col = 'diag\_1\_' + selected\_diag\_1.replace(' ', '').replace('/', '') # Example: 'Diabetes' -> 'diag\_1\_Diabetes'

if diag\_1\_col in data:

data[diag\_1\_col] = 1

diag\_2\_col = 'diag\_2\_' + selected\_diag\_2.replace(' ', '').replace('/', '') # Example: 'Digestive' -> 'diag\_2\_Digestive'

if diag\_2\_col in data:

data[diag\_2\_col] = 1

diag\_3\_col = 'diag\_3\_' + selected\_diag\_3.replace(' ', '').replace('/', '') # Example: 'Injury' -> 'diag\_3\_Injury'

if diag\_3\_col in data:

data[diag\_3\_col] = 1

# Ensure the column order matches your model\_features list exactly!

# This list MUST match the `model\_features` list in your TabPy script.

column\_order = [

'age\_[50-60)', 'age\_[60-70)', 'age\_[70-80)', 'age\_[80-90)', 'age\_[90-100)',

'diabetes\_med\_yes', 'has\_procedure\_performed',

'has\_outpatient\_visit', 'has\_inpatient\_visit', 'has\_emergency\_visit',

'diag\_1\_Diabetes', 'diag\_1\_Digestive', 'diag\_1\_Injury', 'diag\_1\_Musculoskeletal', 'diag\_1\_Other', 'diag\_1\_Respiratory',

'diag\_2\_Diabetes', 'diag\_2\_Digestive', 'diag\_2\_Injury', 'diag\_2\_Musculoskeletal', 'diag\_2\_Other', 'diag\_2\_Respiratory',

'diag\_3\_Diabetes', 'diag\_3\_Digestive', 'diag\_3\_Injury', 'diag\_3\_Musculoskeletal', 'diag\_3\_Other', 'diag\_3\_Respiratory',

'medical\_specialty\_Emergency/Trauma', 'medical\_specialty\_Family/GeneralPractice',

'medical\_specialty\_InternalMedicine', 'medical\_specialty\_Missing', # Include if applicable

'medical\_specialty\_Other', 'medical\_specialty\_Surgery'

# ... add any other features in the correct order

]

input\_df = pd.DataFrame([data], columns=column\_order) # Use [data] to create a single row DataFrame

# Print the input DataFrame for debugging

print(f'Input DataFrame to model:\n{input\_df}')

# Extract values from the DataFrame into a list for JSON serialization

input\_list = input\_df.values.tolist()

# Call the deployed TabPy function with the list

result = tabpy.query('ReadmissionRiskPredictor', input\_list)['response']

print(f'Prediction result: {result}')

return result

",

// Now, list the Tableau fields and parameters that correspond to \_arg1, \_arg2, etc.

// The order here MUST match the order you used to access \_argN above.

[Select Age Group],

[Has Inpatient Visit],

[Has Outpatient Visit],

[Has Emergency Visit],

[Has Procedure Performed],

[Prescribed Diabetes Med],

[Select Medical Specialty],

[Select Primary Diagnosis],

[Select Secondary Diagnosis],

[Select Tertiary Diagnosis]

)

1. Click **OK**.

## 

## Step 8: Use the Calculated Field on a Worksheet

The Predicted Readmission Risk field is now in your Tableau Data pane. You can use it like any other measure.

* Drag it to the **Rows** or **Columns** shelf.
* Place it on **Color** to create a heatmap.
* Add it to a tooltip to show the prediction when you hover over a mark.

This allows you to visualize and analyze your Python model's predictions directly within a Tableau dashboard.