

# Linear Regression using Tableau and TabPy module

Author: Aditya Wresniyandaka

Pre-requisite: Tableau 2018 or 2019 and Python 3 installed. A convenient option is via Anaconda installation.

The steps described in this document are for Microsoft Windows 10.

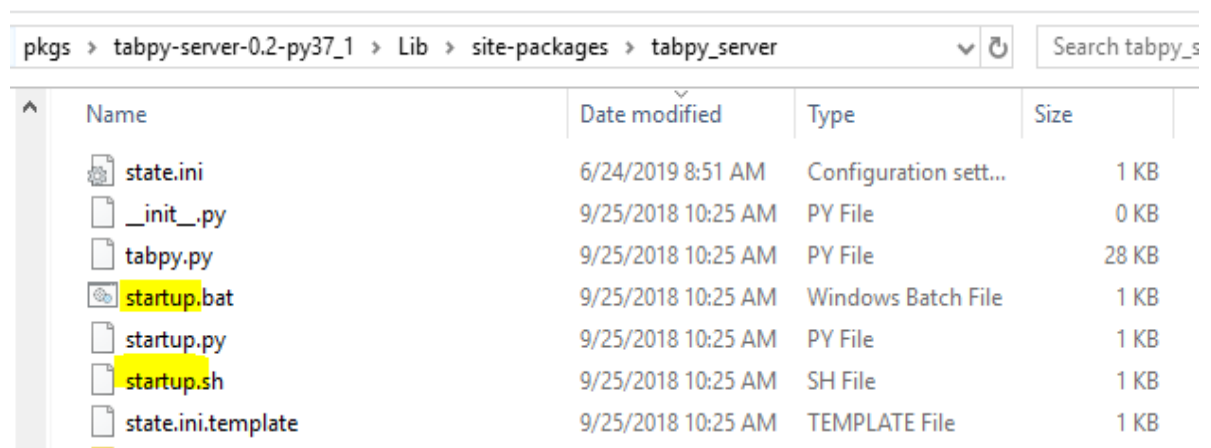
## Configuring TabPy Server and connecting from Tableau Desktop

Add/install TabPy (Tableau and Python) integration module using Anaconda:

1. Open an Anaconda prompt
2. Issue the command: **conda install -c anaconda tabpy-server**, and follow the prompt. The installation will take a while to complete.
3. Find the path to the **Tabpy server startup program** on your machine. It can be .bat or .sh depending on your Operating System.

For example:

C:\Users\atwresn\AppData\Local\Continuum\anaconda3\pkgs\tabpy-server-0.2-py37\_1\Lib\site-packages\tabpy\_server



pkgs > tabpy-server-0.2-py37_1 > Lib > site-packages > tabpy_server					Search tabpy_s
^	Name	Date modified	Type	Size	
	state.ini	6/24/2019 8:51 AM	Configuration sett...	1 KB	
	__init__.py	9/25/2018 10:25 AM	PY File	0 KB	
	tabpy.py	9/25/2018 10:25 AM	PY File	28 KB	
	startup.bat	9/25/2018 10:25 AM	Windows Batch File	1 KB	
	startup.py	9/25/2018 10:25 AM	PY File	1 KB	
	startup.sh	9/25/2018 10:25 AM	SH File	1 KB	
	state.ini.template	9/25/2018 10:25 AM	TEMPLATE File	1 KB	

4. You can see the status and port number where the tabpy server is listening on your machine:

```
(base) C:\Users\adity\Anaconda3\pkgs\tabpy-server-0.2-py36_1\Lib\site-packages\tabpy_server>startup.bat
Could not find conda environment: Tableau-Python-Server
You can list all discoverable environments with `conda info --envs`.

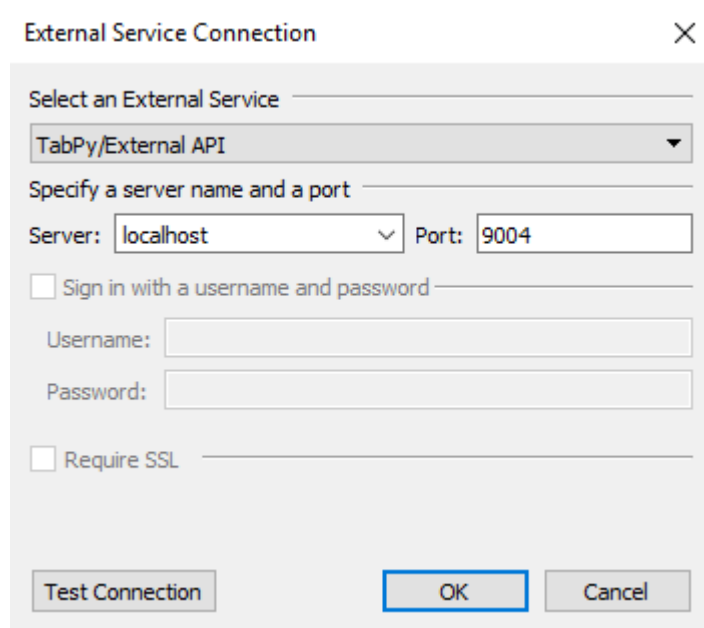
Initializing TabPy...
Done initializing TabPy.
Web service listening on port 9004
```

You can also check from your browser, by typing this:

<http://localhost:9004>

You should see the Tableau logo.

5. From Tableau desktop, make a connection to the TabPy server:  
Go to Help – Settings and Performance – Manage External Service Connection:



Note that Tableau 2019.2 supports Secure TabPy connections.

### Tableau Desktop – sample calculated fields that use TabPy

To utilize an external service, Tableau provides four methods:

- SCRIPT\_BOOL()  
SCRIPT\_INT()  
SCRIPT\_REAL()  
SCRIPT\_STR()

You can find the documentation on each of these functions in the online help:

[https://onlinehelp.tableau.com/current/pro/desktop/en-gb/functions\\_functions\\_tablecalculation.htm#scriptbool](https://onlinehelp.tableau.com/current/pro/desktop/en-gb/functions_functions_tablecalculation.htm#scriptbool)

Note that for Python, the argument uses a leading underscore. For example, `_arg1`, `_arg2`. These arguments are used to pass the values from Tableau to Python.

To illustrate how we can integrate Tableau and Python, we will use the Sample Superstore data source that is already embedded in Tableau Desktop. We will only use three fields: Customer Name for the

dimension, Sales and Profit for the measures. We will do an analysis and machine learning prediction using a simple Linear Regression:  $y = a + bx$  where we want to predict Profit (y) based on Sales (x). a is the Intercept and b is the Coefficient.

### Sample scenarios

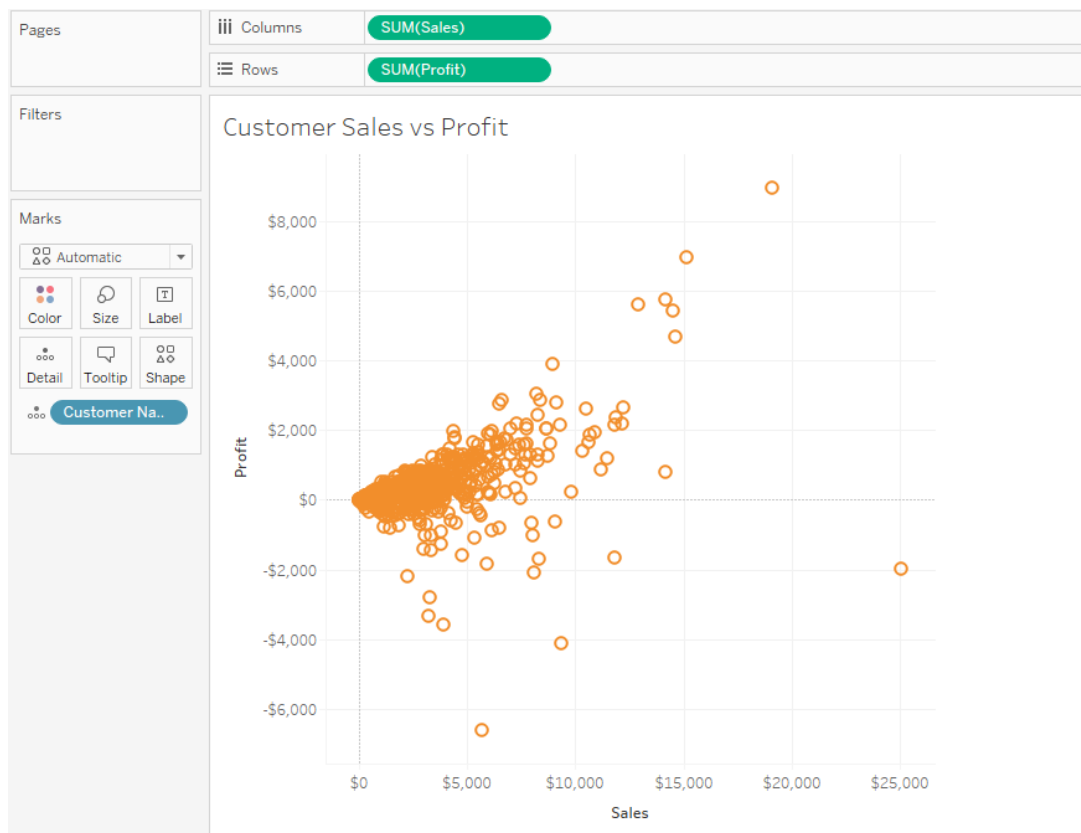
There are two sample scenarios:

1. Plot a fitted line using Linear Regression from the `sklearn.linear_model` package in Python dynamically in Tableau
2. Perform predictive analytics using a model built in Python and deployed in TabPy server. Tableau will call the model with an arbitrary Sales value and the model will produce a predicted Profit value dynamically. The prediction is run in TabPy server.

### Plotting a fitted line

In the first scenario, Tableau calls Python real-time to build a model on-the-fly and return the Coefficient and Intercept based on the set of data that is in the visualization.

1. You can start with a simple scatter plot as follows:



2. Calculate the Linear Regression Coefficient and Intercept using Python.  
To calculate the Coefficient, create a calculated field and name it for example **PythonRegressionCoeff**:

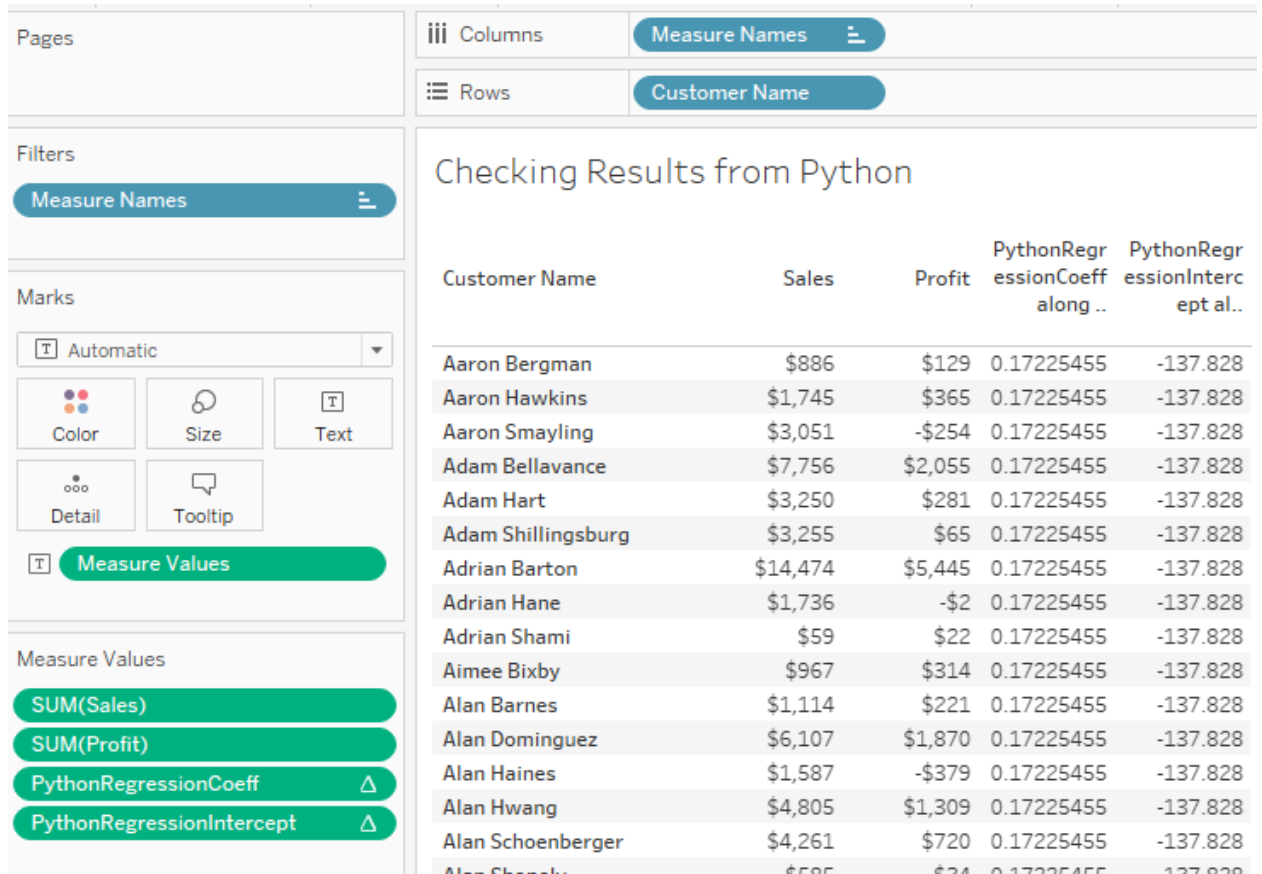
```
SCRIPT_REAL("
import numpy as np
from sklearn.linear_model import LinearRegression
model = LinearRegression()
df_X = np.array(_arg1).reshape(-1,1)
df_Y = np.array(_arg2).reshape(-1,1)
model.fit(df_X, df_Y)
return np.asscalar(model.coef_)
", sum([Sales]),sum([Profit]))
```

In this script, we are passing sum of Sales and Profit to Python via `_arg1` and `_arg2` as lists. We then convert it in Python into an array and reshape into a format that is accepted by the `fit()` method. After that, we build the model and ask Python to return the Coefficient value from the model.

Similar to the above, we create a calculated field for example **PythonRegressionIntercept** with the following code:

```
SCRIPT_REAL("
import numpy as np
from sklearn.linear_model import LinearRegression
model = LinearRegression()
df_X = np.array(_arg1).reshape(-1,1)
df_Y = np.array(_arg2).reshape(-1,1)
model.fit(df_X, df_Y)
return np.asscalar(model.intercept_)
", sum([Sales]),sum([Profit]))
```

Although we pass two lists of values to Python, it should return only two unique numbers – the Coefficient and the Intercept. You can build a visualization to see the results, as follows:

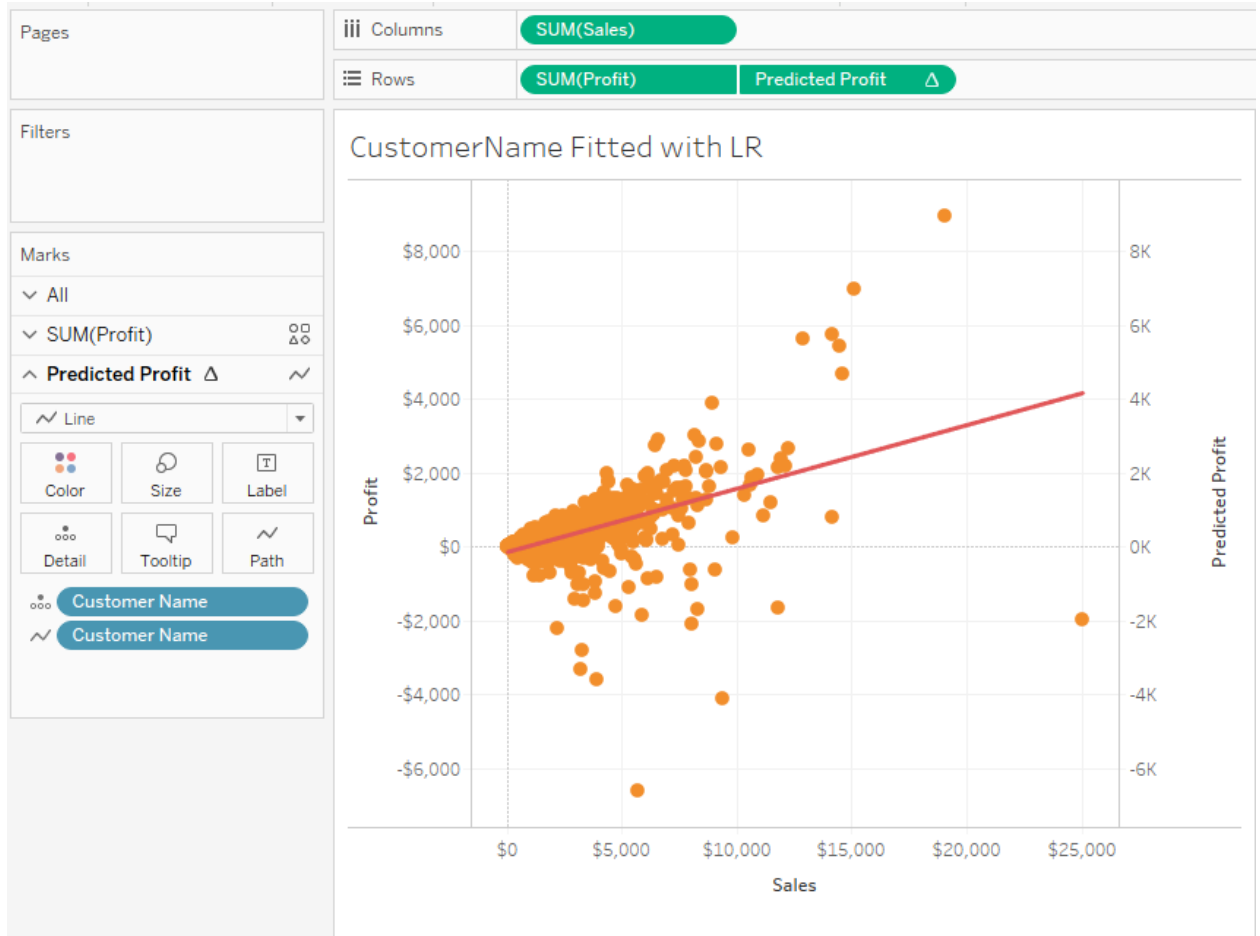


3. Create a calculated field in Tableau to show the Predicted values based on Sales values:

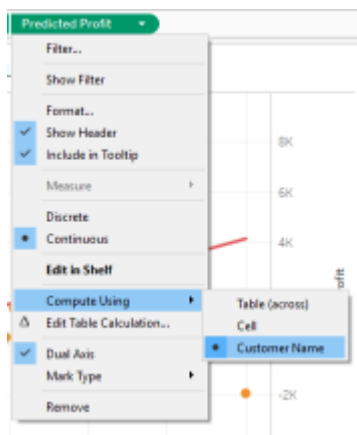
Predicted Profit

```
[PythonRegressionIntercept] + (Sum([Sales]) * [PythonRegressionCoeff])
```

4. Update the scatter plot to include the fitted line (the value of Predicted Profit) using Dual Axis – Synchronize Axis, based on Customer Name:

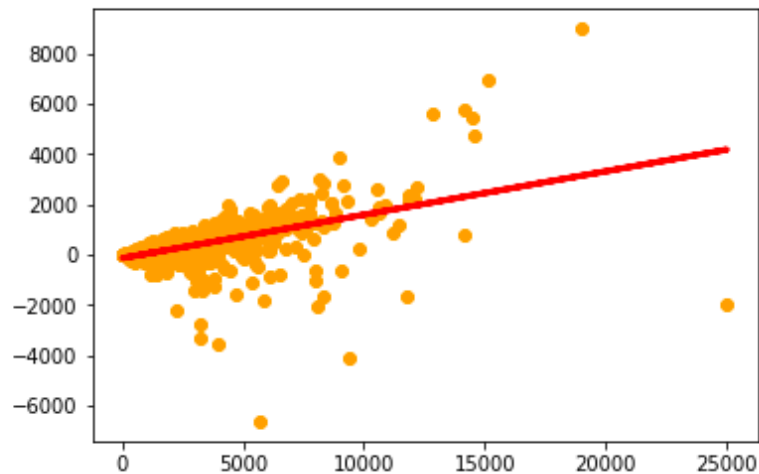


Note that the Predicted Profit needs to be calculated using Customer Name:



As a reference, in Python we get the same plot:

```
## Plot the data with the fitted line  
import matplotlib.pyplot as plt  
plt.scatter(sales['Sales'], sales['Profit'], color='orange')  
plt.plot(sales['Sales'], sales['Predicted profit'], color='red', linewidth=3)  
plt.show()
```



#### Predictive Analytics using Tableau and Python via TabPy

In the previous scenario, we call Python to build the model on-the-fly so if there is a new set of data, Tableau will render the latest Coefficient and Intercept values.

In this scenario, we build the model in Python, wrap the call to run a prediction in a function, and then deploy the function to the TabPy server. Tableau will call the function by passing a Sales number and TabPy will return the predicted profit.

The steps in Python:

1. Read a CSV that contains the Customer Name, Sales and Profit data.

```
import numpy as np  
import pandas as pd
```

```
sales = pd.read_excel("C:\\Users\\aditya\\Documents\\TableauWork\\Customer_SuperStore.xlsx")
```

2. Build the model using `linear_model` in the `sklearn` package

```

## Build a model using Linear Regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
df_X = pd.DataFrame(sales['Sales'].values)
df_Y = pd.DataFrame(sales['Profit'].values)
model.fit(df_X, df_Y)

```

3. Create a function that wraps the prediction

```

## Create a function that wraps the prediction
def predictProfit(sales):
    return model.predict(sales).tolist()

```

4. Make a connection to the tabpy server using the tabpy\_client module:

```

## Make a connection to the tabpy server
import tabpy_client
connection = tabpy_client.Client("http://localhost:9004")

```

5. Deploy the function and model to the tabpy\_server:

```

## Deploy the model to the tabpy server
connection.deploy('PredictProfit', predictProfit, 'The function to predict profit based on sales number.')

```

In Tableau, you will create a calculated field to call the function deployed in TabPy server. The call and returned values vary depending on the function and model that you build in Python.

In the following example the return value is in a JSON format, so we will extract the 'response' and the first item in it. We will use a parameter to capture the value that the user will enter.



Steps in Tableau Desktop:

1. Create a parameter:

Edit Parameter [Enter Sales] X

Name:  Comment >>

Properties

Data type:

Current value:

Display format:

Allowable values: ☐ All ☐ List ☒ Range

Range of values

☒ Minimum:

☒ Maximum:

☐ Step size:

2. Create a calculated field to store the value in the parameter so we can pass it to TabPy:

---

`[Enter Sales]`

3. Create a calculated field to call the function and capture the returned value. Notice that we pass a single value to the tabpy query.

---

Results are computed along Table (across).

```
SCRIPT_REAL("
return tabpy.query('PredictProfit',_arg1[0])['response'][0]", [MySales])
```

4. You can test the interaction between Tableau and TabPy using a simple worksheet as follows:

### What if Sales and Predicted Profit

My Sales	Predicted Profit
My Sales: 2000	\$206.68

Enter Sales

\$2,000.00

As you enter a value in the parameter, notice that the number in Predicted Profit changes.

Some samples on how you can build an interactive dashboard using this integration:

