Ambiente virtuale dove eseguire il codice Python per PowerBI Desktop

1. Installazione di Python sulla macchina
2. Installazione del virtualizzatore dell'ambiente di esecuzione nel quale installare le librerie richieste

```
$ py -m pip install --user virtualenv
```

3. Creazione dell'ambiente virtuale pbi_py

```
$ py -m venv pbi_py
```

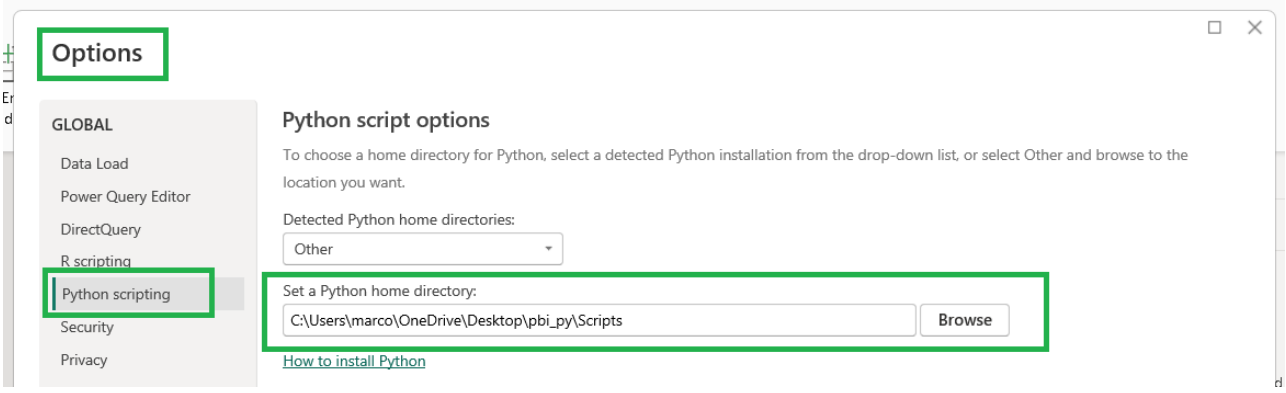
4. Attivazione

```
$ .\ pbi_py \Scripts\activate
```

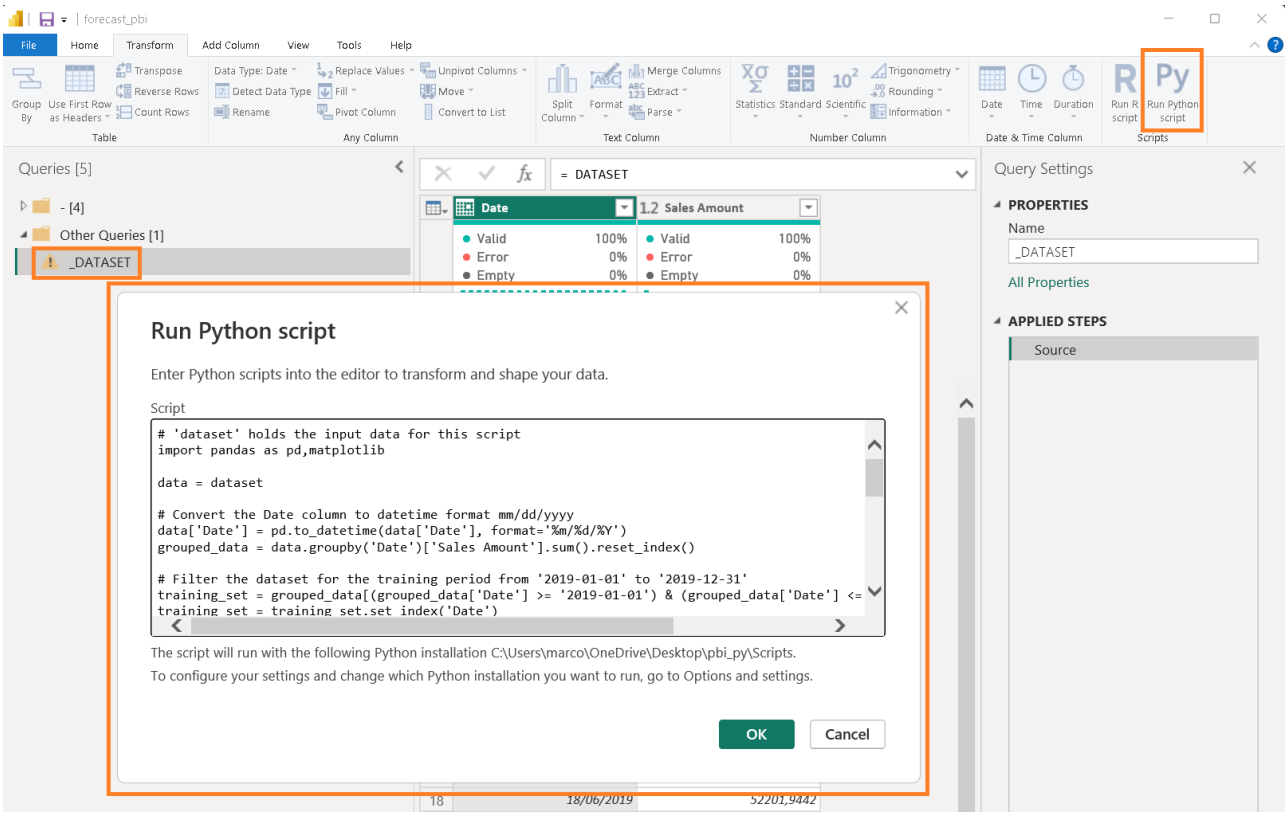
5. Installazione delle librerie necessarie

```
$ pip install pandas
$ pip install numpy
$ pip install statsmodels
$ pip install matplotlib
```

6. Far puntare l'ambiente di esecuzione in Power BI Desktop al ambiente virtuale appena creato.
   L'immagine mostra come farlo attraverso il pannello delle impostazioni.



7. In Power Query creare uno step per eseguire script Python, ricordandosi di salvare i risultati in un DataFrame. In questo modo allo step successivo sarà possibile espandere la tabella con i risultati del forecast. L'immagine mostra come farlo in Power Query.



8. Disattivare l'ambiente virtuale quando il report è concluso

```
$ deactivate
```

Modello HOLT-WINTERS

Creare in Power Query uno *step* per l'esecuzione del codice Python e incollare nella finestra del codice quanto segue.

```
# 'dataset' holds the input data for this script
import pandas as pd,matplotlib

data = dataset

# Convert the Date column to datetime format mm/dd/yyyy
data['Date'] = pd.to_datetime(data['Date'], format='%m/%d/%Y')
grouped_data = data.groupby('Date')['Sales Amount'].sum().reset_index()

# Filter the dataset for the training period from '2019-01-01' to '2019-12-31'
training_set =
grouped_data[(grouped_data['Date'] >= '2019-01-01') \
& \
(grouped_data['Date'] <= '2019-12-31')]
training_set = training_set.set_index('Date')

# Reference time period for Forecasting
forecast_start = '2020-01-01'
forecast_end = '2020-03-31'

# Filter the original dataset for the test period ('2020-01-01' to '2020-03-31')
to compare against the forecast
test_set =
grouped_data[(grouped_data['Date'] >= '2020-01-01') \
& \
(grouped_data['Date'] <= '2020-03-31')].set_index('Date')

from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Exponential Smoothing model using the Holt-Winters Seasonal Method with
specified parameters
model_hw = ExponentialSmoothing(
    training_set['Sales Amount'],
    trend='add',
    seasonal='add',
    seasonal_periods=91, # defined lowering the SSE
    initialization_method="estimated"
).fit(optimized=True)

# Forecast the next 90 days with the specified Holt-Winters model
forecast = model_hw.forecast(steps=90)

# Create a dataframe to memorize data for PowerBI
forecast = pd.DataFrame(forecast).reset_index()
```

Modello SARIMA

Creare in Power Query uno *step* per l'esecuzione del codice Python e incollare nella finestra del codice quanto segue.

```
# 'dataset' holds the input data for this script
import pandas as pd,matplotlib

data = dataset

# Convert the Date column to datetime format mm/dd/yyyy
data['Date'] = pd.to_datetime(data['Date'], format='%m/%d/%Y')
grouped_data = data.groupby('Date')['Sales Amount'].sum().reset_index()

# Filter the dataset for the training period from '2019-01-01' to '2019-12-31'
training_set =
grouped_data[(grouped_data['Date'] >= '2019-01-01') \
& \
(grouped_data['Date'] <= '2019-12-31')]
training_set = training_set.set_index('Date')

# Reference time period for Forecasting
forecast_start = '2020-01-01'
forecast_end = '2020-03-31'

# Filter the original dataset for the test period ('2020-01-01' to '2020-03-31')
to compare against the forecast
test_set =
grouped_data[(grouped_data['Date'] >= '2020-01-01') \
& \
(grouped_data['Date'] <= '2020-03-31')].set_index('Date')

from statsmodels.tsa.statespace.sarimax import SARIMAX

# Define SARIMA parameters (this is an example for the evaluation consider more
sophisticated methods)
order = (1, 1, 1)
seasonal_order = (1, 1, 0, 90)

# Fit the SARIMA model with the specified parameters
model = SARIMAX(training_set, order=order, seasonal_order=seasonal_order,
enforce_stationarity=False, enforce_invertibility=False)
model_fit = model.fit(disp=False)

# Forecasting
forecast = model_fit.predict(start=forecast_start, end=forecast_end)

# Create a dataframe to memorize data for PowerBI
forecast = pd.DataFrame(forecast).reset_index()
```