Mobile Phone Decision Tree Classifier- Documentation

This document contains guidance on **Scripting a Python Model into Power BI**, using the mobile phone data; a decision tree model to predict the mobile phone price band is developed inside Power BI using Python script, with the resulting model used to populate Power BI visual reports. This example can be reused for any similar predictive scenario that you want to embed in Power BI to share reports in the business.

TIP : Power BI is highly compatible with Pandas. If you cannot find an array or series that you have processed, try combining it with other columns and turn it into a Data Frame. You can also create a one column data frame, convert a series or array to a data frame.

- https://pandas.pydata.org/docs/reference/api/pandas.Series.to_frame.html

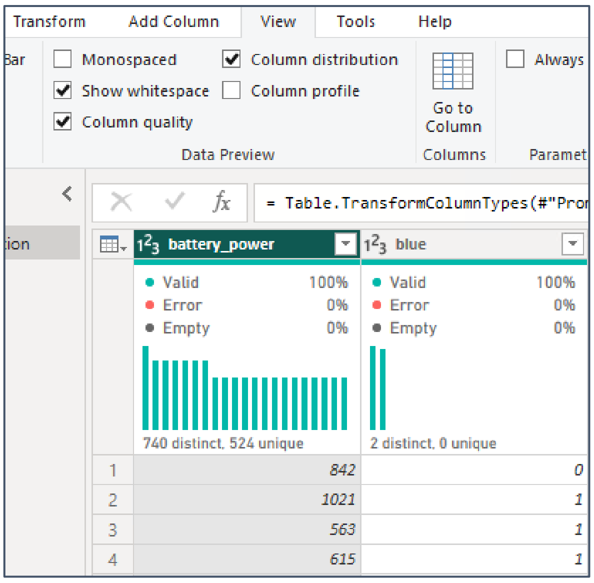
- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

Setup for running Python in Power BI:

- Running Python scripts in Power BI requires some initial setup – this link includes the path configuration and a simple test Python script as a data source into the Power BI workbook

- https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-Python-scripts

- Ensure that required Python libraries [numpy, pandas, scikit-learn, seaborn, matplotlib are installed on your Python instance]
- Connect to the data source in Power BI **phone_classification.xlsx**
- Select Transform Data to access Power Query Editor
- Access and view Column Quality and Distribution (features can be switched on in the View toolbar). Take note of the feature distribution, data gaps and any data processing requirements. Identify the label column (price range) which you will aim to predict using your classification model. This label column is your ‘y’ in machine learning notation.



☐ Monospaced
✓ Show whitespace
✓ Column quality
✓ Column distribution
☐ Column profile
☐ Always

```
= Table.TransformColumnTypes(#"Pro
```

	battery_power	blue
	● Valid 100%	● Valid 100%
	● Error 0%	● Error 0%
	● Empty 0%	● Empty 0%
	740 distinct, 524 unique	2 distinct, 0 unique
1	842	0
2	1021	1
3	563	1
4	615	1

Fig 1. Using the View Tab in Power Query supports Data Evaluation

- Perform necessary data cleaning or data type transformations as appropriate to your data set using Power BI native features.
- When satisfied with overall quality, duplicate the data and label the duplicate *DataFrames*. This is where you will primarily be processing data using Python scripts to create the data for your report.
- To begin using Python to process your data, select **Run Python script**

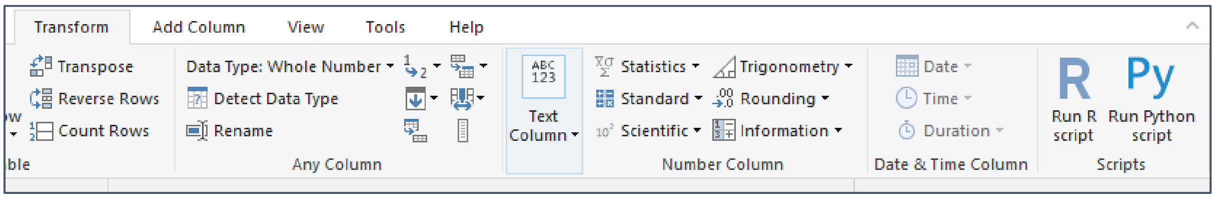
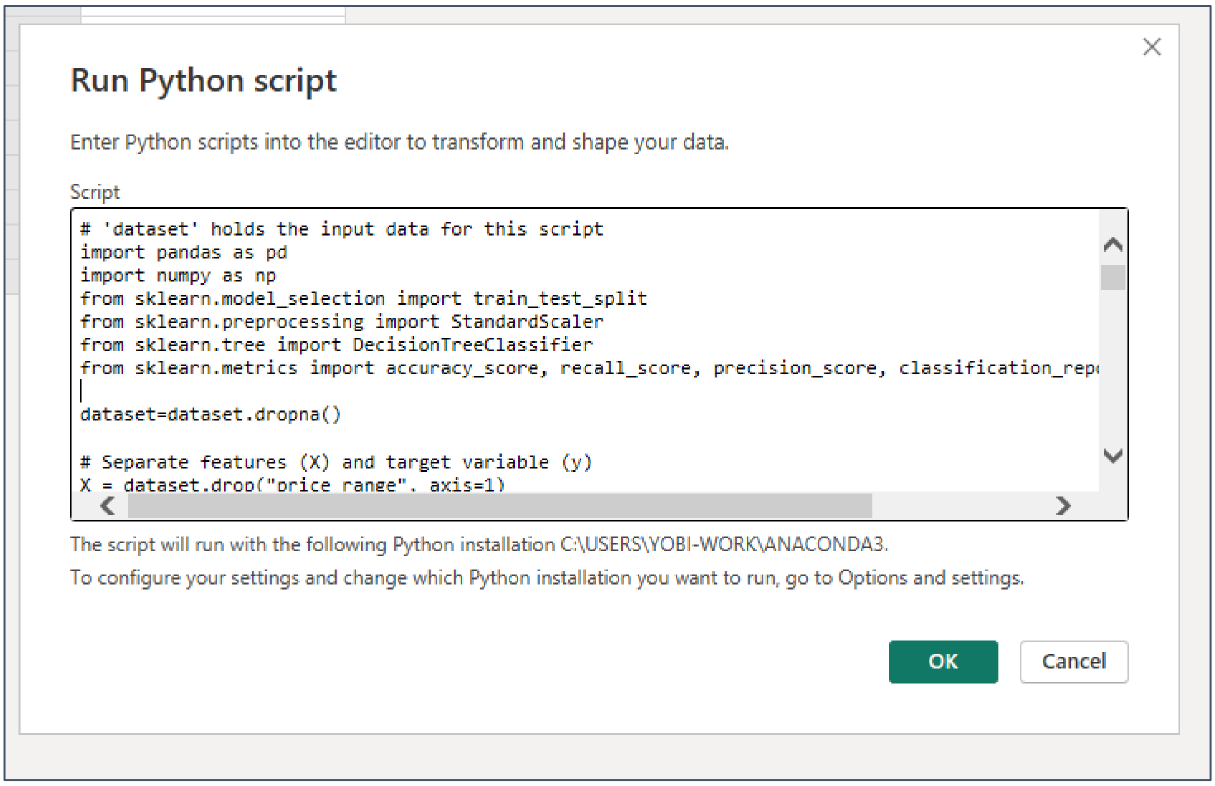


Fig 2. Python processing in Power Query

Data Processing with Python

- Assuming you already have a functioning model in Python/ Jupyter notebook, you can transfer much of the data processing model and evaluation script in your Jupyter notebook to this input box before selecting OK to run the script, noting the correct location on your computer where the Python application is installed.


Run Python script
Enter Python scripts into the editor to transform and shape your data.
Script

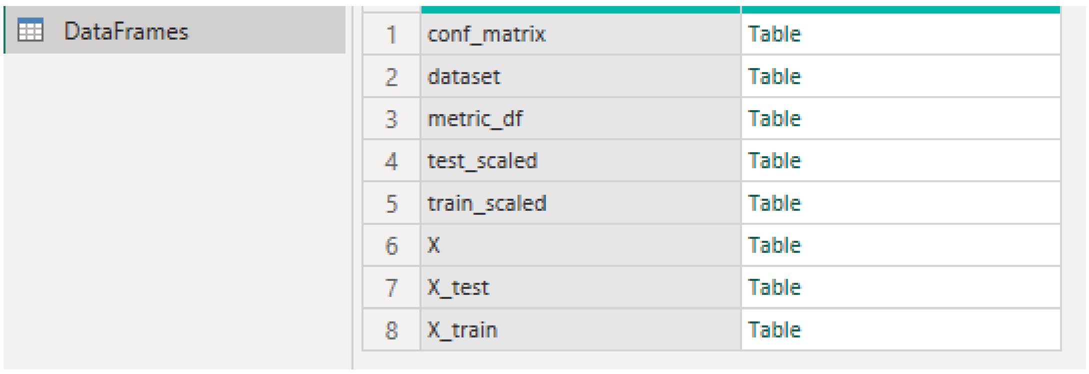
```
# 'dataset' holds the input data for this script
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, recall_score, precision_score, classification_rep

dataset=dataset.dropna()

# Separate features (X) and target variable (y)
X = dataset.drop("price range", axis=1)
```

The script will run with the following Python installation C:\USERS\YOBI-WORK\ANACONDA3.
To configure your settings and change which Python installation you want to run, go to Options and settings.

Fig 3. Python Script, run as a single block. Adapted code from a Python Notebook

- The script once run should result in a query in Power BI which creates multiple data frames indicating the outputs of the Python you have run.
- Following from the above, these are the data frames that you require in your workbook as Power BI queries to complete your report:
    - dataset (entire dataset is clean and validated)
    - train (scaled X_train and y_train_reset combined in a dataframe)
    - test (scaled X_test and y_test_reset combined in a dataframe)
    - most_informative_features (a simple attribute derived from the model)
    - conf_matrix (y_pred and y_test combined for evaluation)
    - Accuracy, prediction, recall (Model performance metrics combined in a df)

- Once you have completed the data processing, right click the target tables to save them as new queries in your workbook



1	conf_matrix	Table
2	dataset	Table
3	metric_df	Table
4	test_scaled	Table
5	train_scaled	Table
6	X	Table
7	X_test	Table
8	X_train	Table

Fig 4. Desired Output of **Dataframes**

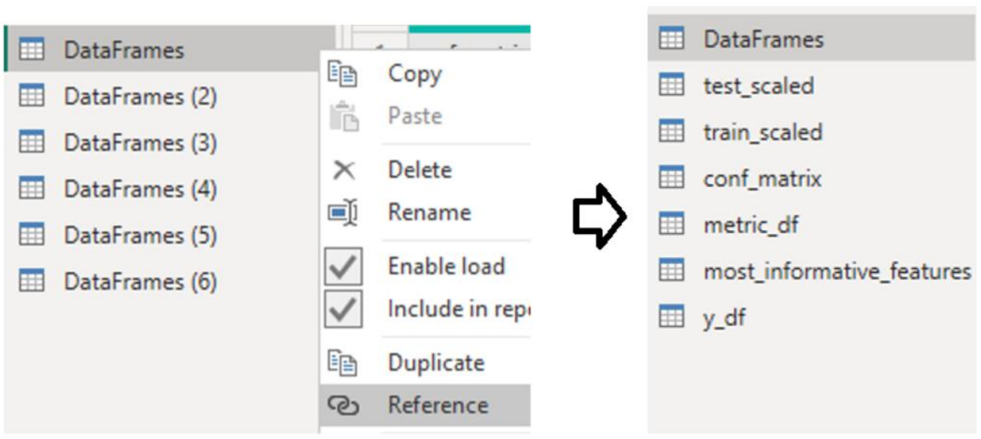


Fig 5. Reference the main Data Frame table, rename and select the corresponding table

Visualisations with Python

Once complete and you are satisfied with the data frames you have processed, go into the main Power BI Desktop report view and begin visualising the model and its performance metrics specifically using the Py button under Visualization

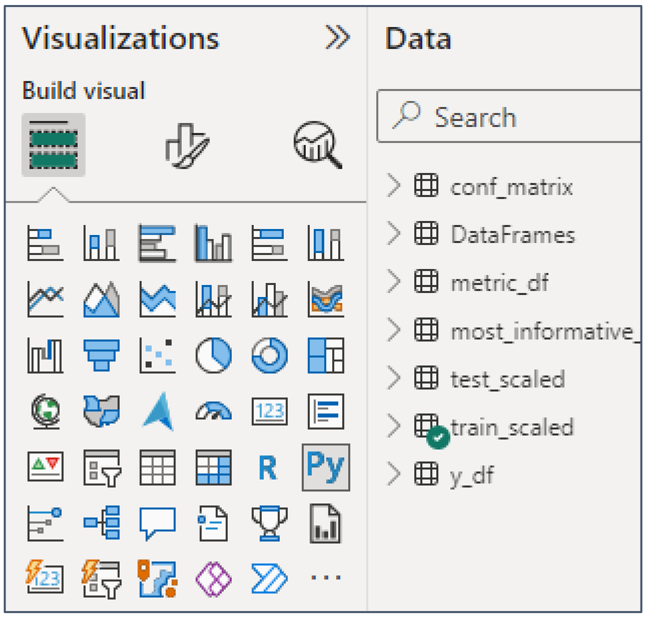
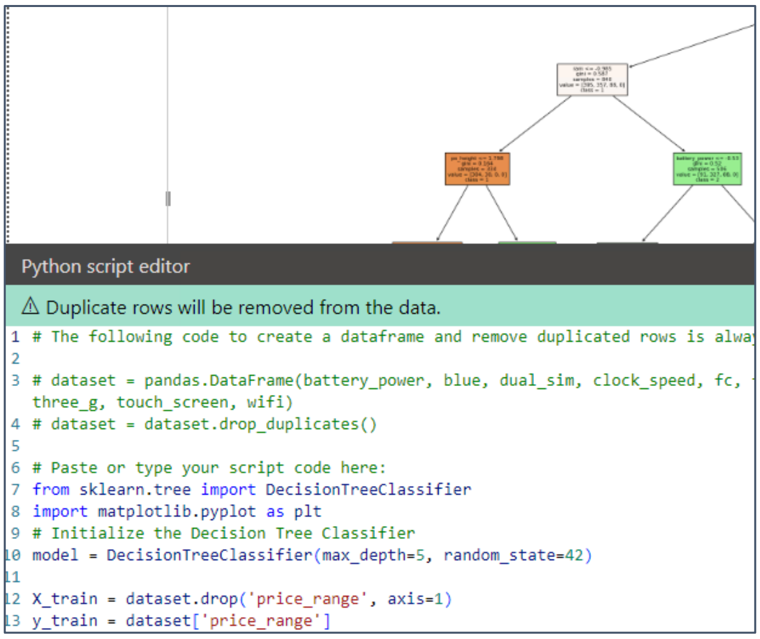


Fig 6. Select the Py button for Python visuals. Later, for visualising the model metrics you may choose to utilise a simple Power BI table rather than Python visuals.

- Adapt your Python visualisation code from your notebook for the Power BI Script editor. **Read the green Python script editor box** in Power BI carefully to understand how to do this.

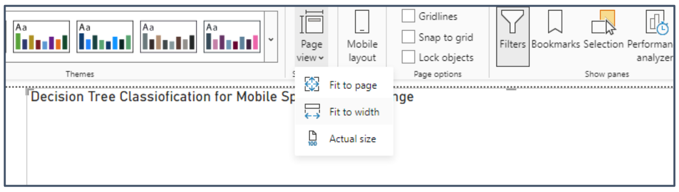


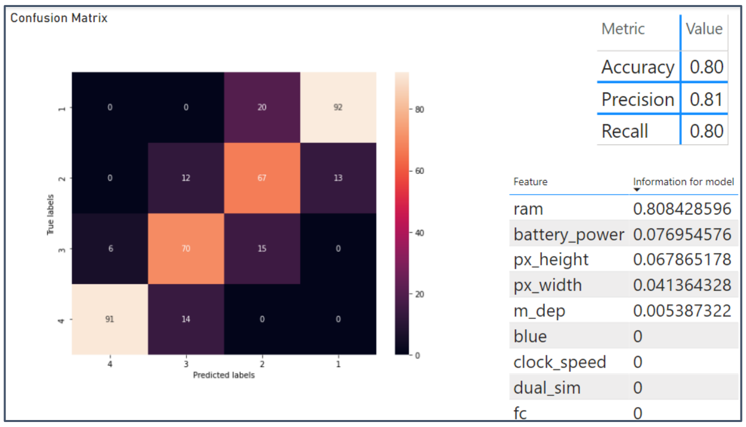
```
# The following code to create a dataframe and remove duplicated rows is alway

# dataset = pandas.DataFrame(battery_power, blue, dual_sim, clock_speed, fc, 
three_g, touch_screen, wifi)
# dataset = dataset.drop_duplicates()

# Paste or type your script code here:
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
# Initialize the Decision Tree Classifier
model = DecisionTreeClassifier(max_depth=5, random_state=42)

X_train = dataset.drop('price_range', axis=1)
y_train = dataset['price_range']
```

Fig 7. Python Script Editor will automatically manipulate your data.

Effective Communication

Experiment with *Figsize*, *page view* and *font size* to create a Decision Tree that is legible, even if you must zoom in to see what's written in the nodes. It is a dense visualisation and if we were to reduce the depth even further, we would have an overgeneralised model.done





Metric	Value
Accuracy	0.80
Precision	0.81
Recall	0.80

Feature	Information for model
ram	0.808428596
battery_power	0.076954576
px_height	0.067865178
px_width	0.041364328
m_dep	0.005387322
blue	0
clock_speed	0
dual_sim	0
fc	0

Fig 8. Report page view (top), example of Model metrics in both Power BI Table and Python Script (Seaborn)