HOW TO EMBED A DECISION TREE IN PBI

INTRODUCTION

this presentation provides guidance on Scripting a Python Model into PowerBI, using the mobile phone data; a decision tree model to predict the mobile phone price band is developed inside PowerBI using Python script, with the resulting model used to populate PowerBI visual reports.

This example can be reused for any similar predictive scenario that you want to embed in PowerBI to share reports in the business

SOURCE DATA AND OBJECTIVE

- data set contains 2000 mobile phones, and a variety of their characteristics
- we want to predict the price range (0-3)
- this is a multi class classifier problem
- we will use a decision tree model to make the prediction

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
2	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	0	0	1	1
3	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1	1	0	2
4	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1	1	0	2
5	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1	0	0	2
6	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1	1	0	1
7	1859	0	0.5	1	3	0	22	0.7	164	1	7	1004	1654	1067	17	1	10	1	0	0	1
8	1821	0	1.7	0	4	1	10	0.8	139	8	10	381	1018	3220	13	8	18	1	0	1	3
9	1954	0	0.5	1	0	0	24	0.8	187	4	0	512	1149	700	16	3	5	1	1	1	0
10	1445	1	0.5	0	0	0	53	0.7	174	7	14	386	836	1099	17	1	20	1	0	0	0
11	509	1	0.6	1	2	1	9	0.1	93	5	15	1137	1224	513	19	10	12	1	0	0	0
12	769	1	2.9	1	0	0	9	0.1	182	5	1	248	874	3946	5	2	7	0	0	0	3
13	1520	1	2.2	0	5	1	33	0.5	177	8	18	151	1005	3826	14	9	13	1	1	1	3
14	1815	0	2.8	0	2	0	33	0.6	159	4	17	607	748	1482	18	0	2	1	0	0	1
15	803	1	2.1	0	7	0	17	1	198	4	11	344	1440	2680	7	1	4	1	0	1	2

PYTHON SETUP

ensure you know where the python application is run on your computer - you will need the path for the powerbi set up

confirm that pandas, scikit-learn, seaborn, matplotlib are installed on your python environment

POWERBI SETUP

Running python scripts in PowerBI requires some initial setup

this link

includes the path configuration and a simple test python script as a data source into the powerbi workbook

SUMMARY OF STEPS

Jupyter

In jupyter notebook, using python/pandas, prepare your test and train data sets, build your ML model with sklearn evaluate it, visualise the model, confusion matrix iterate as needed

PBI power query

import your source data, prepare your data frames with a python script transformation step and save the dataframes as separate queries

PBI reports

Python script visualisation, use python to rebuild model, display tree using matplotlib and sklearn.tree, adjust figsize for the display

assemble 2nd report from remaining dataframes and visualise where possible

JUPYTER NOTEBOOK - PYTHON

DATA

import the source data, explore it visually, clean the data as needed, before splitting into train and test data frames. use Scalers for numerical features, one hot encoder for categorical

MODEL

Run the model Decision Tree Classifier and use it to extract most information features

VISUALISE

Using matplotlib and tree.plot_tree graphic, visualise the model

EVALUATION

collect your metrics to assess the model accuracy score recall score precision score , plot a confusion matrix, and -**if binary class** - area under the curve using roc_curve, auc

turn output into dataframes!

PBI supports data frames only you should convert arrays and series to data frames

DATA FRAMES YOU SHOULD BUILD WITH PYTHON

train data frame

test data frame

derived from source data, contains
X features and y label on columns, observations with scaling or encoding as appropriate

Metric	Value
Accuracy	0.8
Recall	0.8
Precision	0.80856

metrics data frame

column A : metric
Column B : value

	1	2	3	4
1	91	14	0	0
2	6	70	15	0
3	0	12	67	13
4	0	0	20	92

confusion matrix data frame

4 cell matrix indicates correct and incorrect classification predictions

DATA FRAMES YOU SHOULD BUILD WITH PYTHON

feat_imp data frame

column of all the features used in the model and the importance value (out of 1)

Feature	importance to model
battery_power	0.076954576
blue	0
clock_speed	0
dual_sim	0
fc	0
four_g	0
int_memory	0
m_dep	0.005387322
mobile_wt	0
n_cores	0
pc	0
px_height	0.067865178

y data frame

derived from source and predicted labels for y, combine the y volumns from the test data with the model predicted y label, using the row index for matching - so you can directly compare actual and predicted labels

Index	Test	Predicted
101	0	0
102	2	2
103	1	1
104	3	3
105	1	1
106	1	2
107	2	2
108	0	0
109	3	2
110	1	1
111	0	0
112	0	1
113	2	2
114	3	3
115	3	2
116	2	2
117	3	3
118	3	3
119	1	1
120	0	0
121	0	0
122	2	1

POWERBI – TRANSFORM/ POWER QUERY

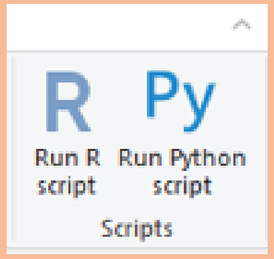
DATA

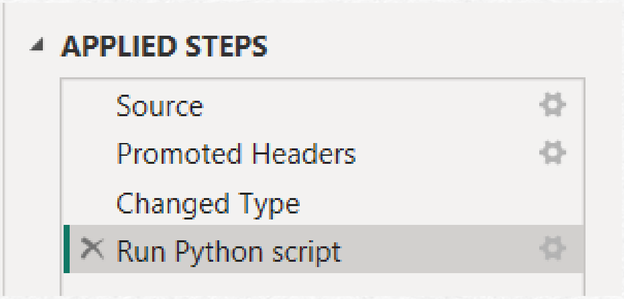
import the source data, explore it visually, clean the data as needed

duplicate the data source, name it ‘dataframes’

PYTHON SCRIPT

import python script from jupyter notebook into a new python transformation step ; creating data frames through the script





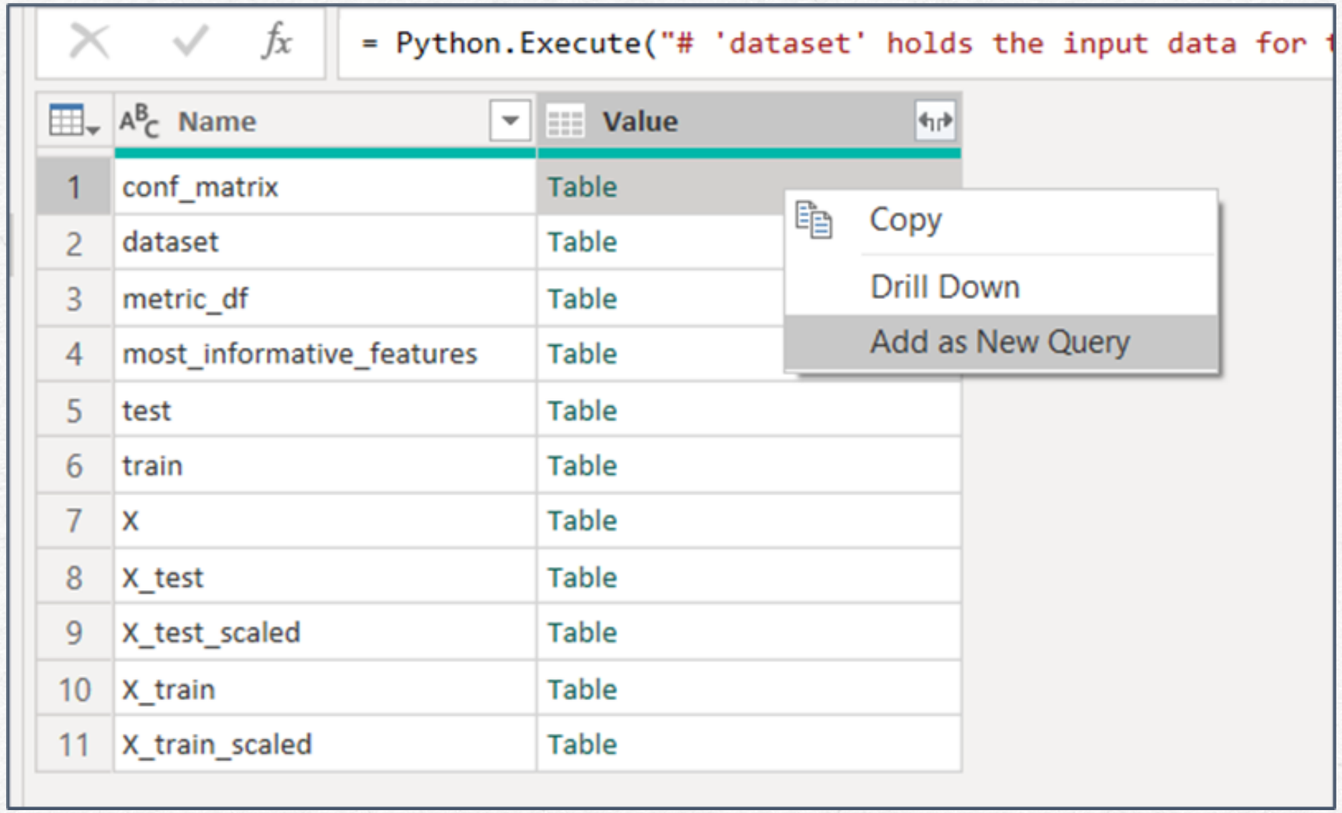
	Name	Value
1	conf_matrix	Table
2	dataset	Table
3	metric_df	Table
4	most_informative_features	Table
5	test	Table
6	train	Table
7	X	Table
8	X_test	Table
9	X_test_scaled	Table
10	X_train	Table
11	X_train_scaled	Table
12	y_df	Table

POWERBI - TRANSFORM/ POWER QUERY

ASSEMBLE QUERIES

output can be exported from the results of the python script step with a right click > add as new

you should have a test df, train df, confusion matrix df, metrics df, most information features df and

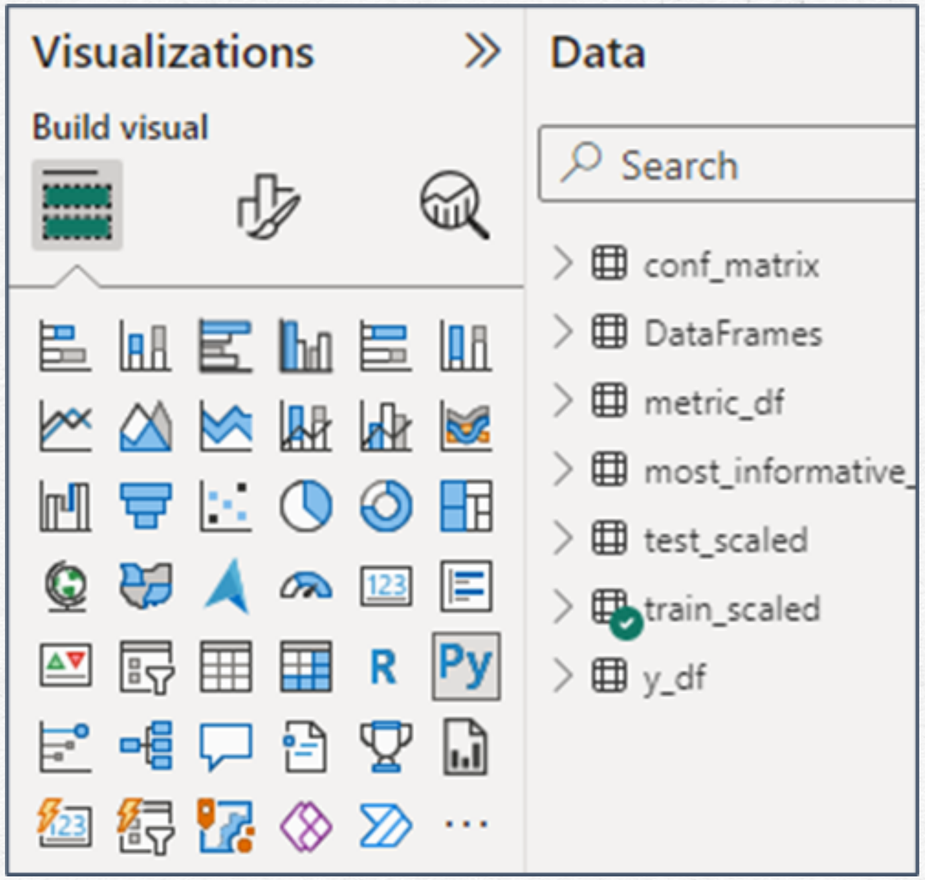


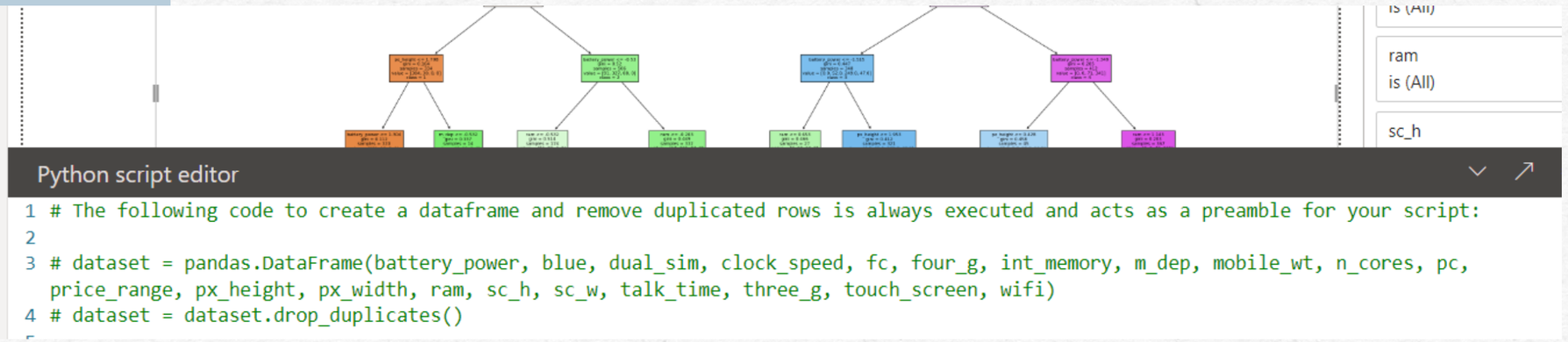
POWERBI VISUALS

DECISION TREE VISUALISATION

once you have visualised the model / tree in jupyter notebook with sklearn.tree.plot_tree and matplotlib.pyplot successfully, you are ready to transfer the query to PBI

- use the Python script visual type
- drag in the features from train df
- recreate the model
- plot the decision tree
- adjust the fig size



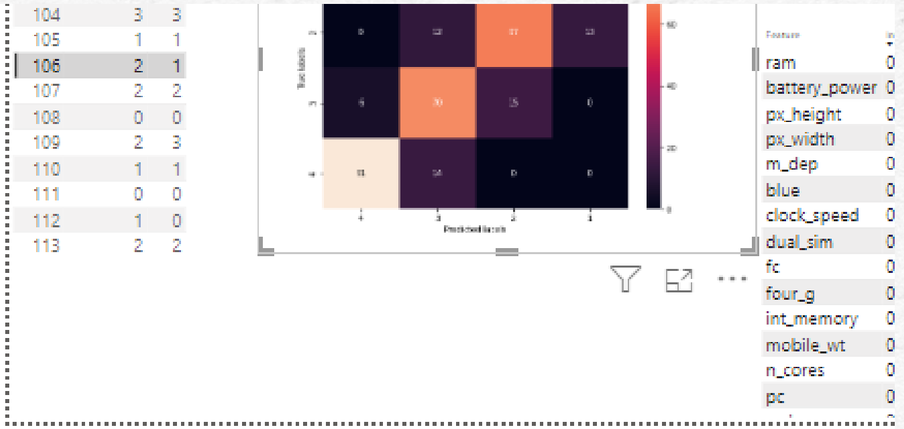


POWERBI VISUALS

CONF MATRIX VISUALISATION

once you have visualised the confusion matrix heatmap in jupyter notebook with seaborn, you are ready to transfer the query to PBI

- use the Python script visual type
- drag in the columns from the conf_matrix data frame
- set up the plot, axes and labels



Python script editor

⚠ Duplicate rows will be removed from the data.

```
# Paste or type your script code here:

import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(dataset, annot=True, fmt='g', ax=ax);  #annot=True to annotate cells, ftm='g' to d

# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.xaxis.set_ticklabels(['4', '3', '2', '1']); ax.yaxis.set_ticklabels(['1', '2', '3', '4'])
plt.show()
```

POWERBI TABLES

to complete your report you should assemble tables which share the metrics, feature importance and predictions as shown, alongside your confusion matrix visual.

Once all the items are arranged on your powerbi report, take the time to work on the layout, adding formatted titles and subheadings to ensure your reports are easy to share and read

Metric	Value
Recall	0.80
Precision	0.81
Accuracy	0.80

Feature	importance of feature
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
four_g	0

TIPS

As you develop your DT Visualisation for your first visual report, experiment with Figsize, with page view and font size to create a Decision Tree that is legible, even if you have 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 will have an overgeneralised model.

ROC visualisation is designed for binary classification problems, not multi class classifiers and throws an error in the case of multi class results. To populate an ROC chart we need to calculate the TPR (true positive rate%) and FPR (false positive rate %) at different thresholds from 0 to 1. sklearn.metrics roc_curve works out these values

You will need to adapt your python visualisation code for the PBI Script editor. Read the green script box instructions in PowerBI carefully to understand how to do this

THAT'S IT!