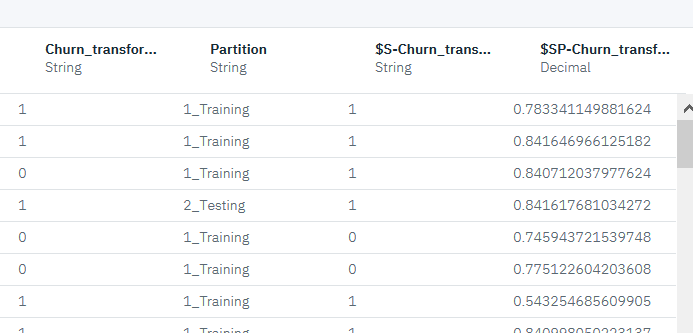
**SCRIPT FOR BOOTCAMP**

1. **VALIDATE WATSON STUDIO DESKTOP INSTALL AND INSTALL NOTEBOOK MODULE**

* Get them to verify they have downloaded the zip file from the repo
* Get them to verify they have installed **WSD** by:
  + Start up WS Desktop
  + Add a Project
  + Adding the CSV file from the repo as a datasource
    - Open the project they created
    - **Add to Project -> Data ->** select the CSV file
  + Prior to starting the lecture part, get them to ensure they have working notebook env:
    - Click on **Add to Projec**t – Notebook and click through – about 3G to download and install – takes about 20 minutes

1. **MODELER**

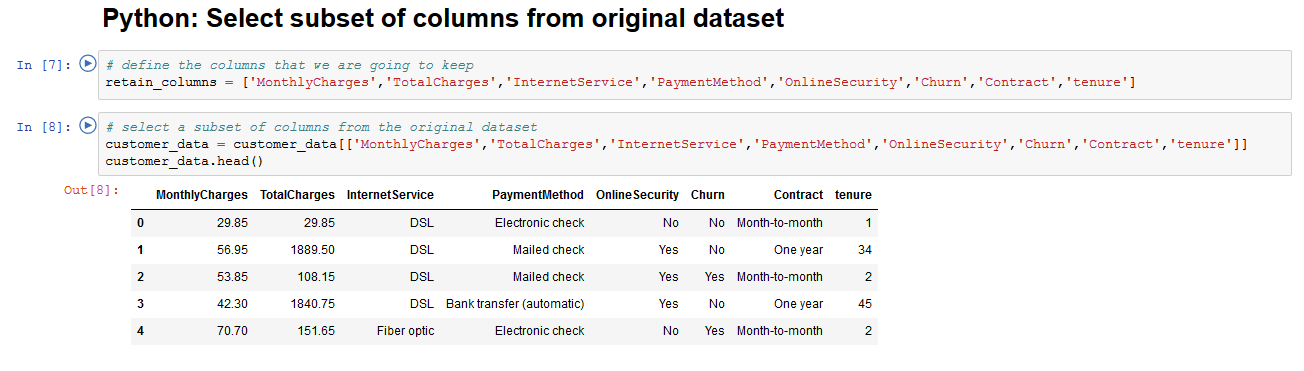
* Open the Project you created in setup step
* Add to Project -> Modeler flow
* Open the new modeler flow:
* Create a Data item and connect to the datasource they created in setup step
* Create a filter item
  + Show Preview
    - Compare preview of initial data node (all cols)
    - With preview of filter node (with just the selected cols)
    - Show the types that are interpreted for the columns in the Preview
  + Show Profile – Churn in particular
  + Show Visualizations – TotalCharge Histogram
    - Churn – pie chart
* Create a filler item
* Create an autoprep item
  + Preview autoprep node
  + NOTE:
    - All numeric
    - Continuous values smoothed
    - Column names changed to “transformed\_”
    - All this done automatically – nothing you need to specify; defaults all logical
* Create a partition item
  + Show the split 70/30
* Create models for SVM and LR
* ***NOTE: ENSURE YOU PREVIEW PARTITION BEFORE ATTEMPTING TO ADD OR RUN MODEL – OTHERWISE WILL NOT WORK***
* Logistic regression
  + Bring LR node onto palette
  + Open LR node
  + Click on “use custom field roles”
  + Select **churn** as target
  + Select all other fields as inputs
  + Save
  + RUN
  + Show PREVIEW on secondary node and highlight the prediction $ columns
  + 
  + NOTE:
    - New node created
* Add analysis node and connect it to the newly created node
  + Update name of analysis node to keep distinct LR from SVM
* RUN ( you need to run first)
  + Then select the first entry to see the results of the prediction
  + Double click on first entry on the list when you open up to see accuracy results
* *It happened that run of analysis didn’t show results – if that happens, just shut down flow, reopen and run again – that will make it work*

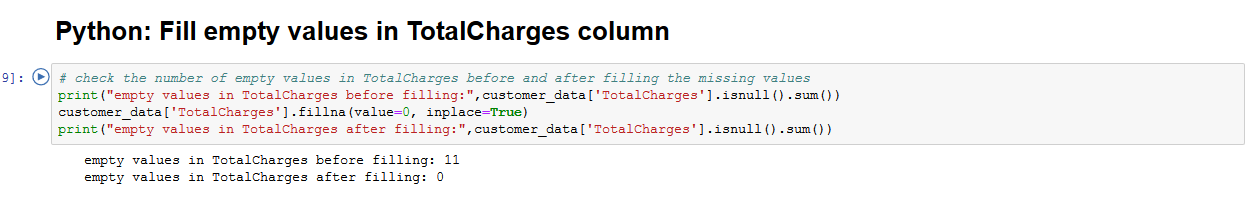
1. **Python**

* Explain what Python is; explain what a notebook is
* Show how to load a Python file into WS
  + From a URL
  + From a
* Show how to open a notebook in WS
  + Show how to open notebook from URL
  + In github repo, click on ipynb file, then click on raw and copy the URL
  + In WS, select open from URL and paste the URL you copied and create
* Show the basic controls in Jupyter
* Show how to execute a cell
* Show a markup cell
* Show how to reset the environment
* Show how to read in a CSV file from a URL
* Show how to read a CSV file from the file system
* Show how to run all the cells in a notebook

Describe the parts of the notebook:

* The libraries that you will need in the notebook
* Os for the file system
* Pandas for dataframes
* Sklearn for ML models
* 
* This notebook works automatically because it’s getting the dataset from a URL in the github repo
* Try to bring the CSV file instead:
  + Then go to the Files section to the right of this notebook and click Insert to code for the data you have uploaded. Choose Insert pandas DataFrame.
  + Put in a line customer\_data = *df\_data\_1*



* First line is a Python list – square bracket with elements of the list separated by commas
* Second line says redefine customer\_data dataframe to contain only the columsn in the list
* And display
* 
* Here we print out a statement that counts the number of null entries in this column
* Then a line that replaces the null entries with 0
* Then print out the number of null entries again



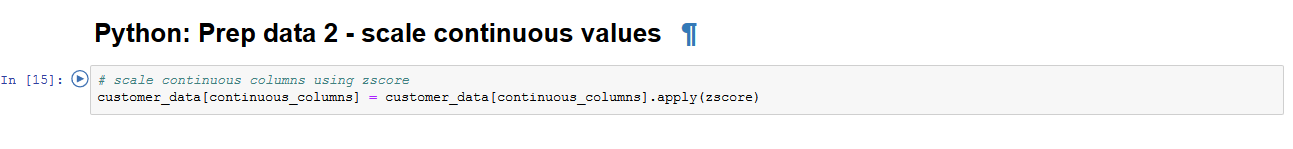
First cell: define two lists, one for the columns that are categorical and one for continuous numerical. Note that tenure is integer but still continuous.

Second cell: a Python function – two arguments passed in – first is a dataframe, second is a list of column names

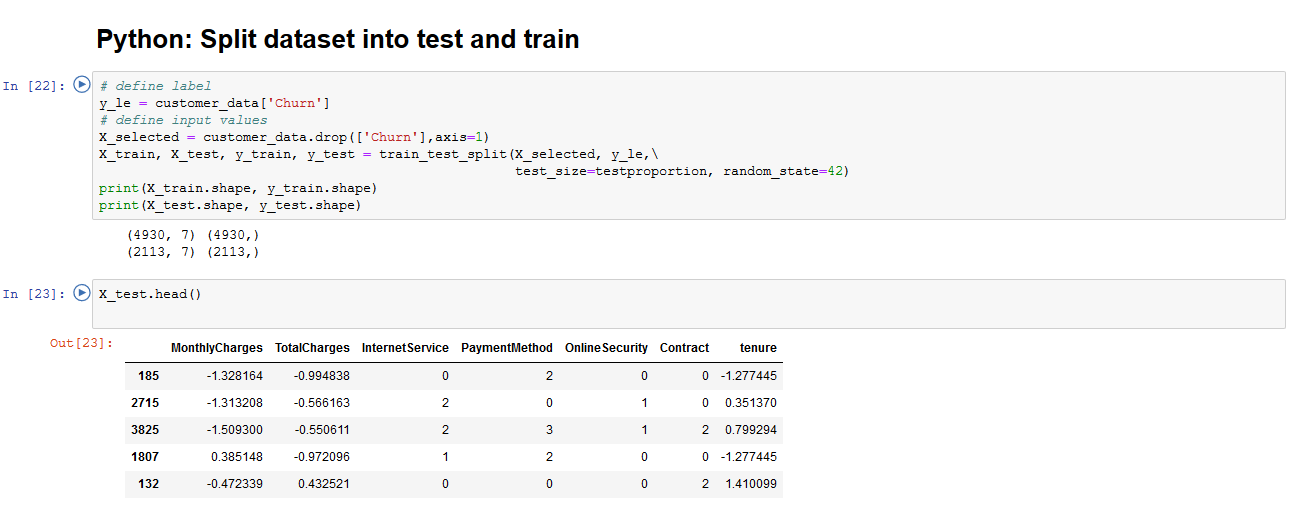
* Scale\_columns
  + For each element of the list, apply the zscore scaling function
* Encode\_columns
  + For each element of the list, apply the labelencoder (which replaces the elements of the dataframe column with integer IDSs)

Third cell: calls encode\_column on the categorical columns

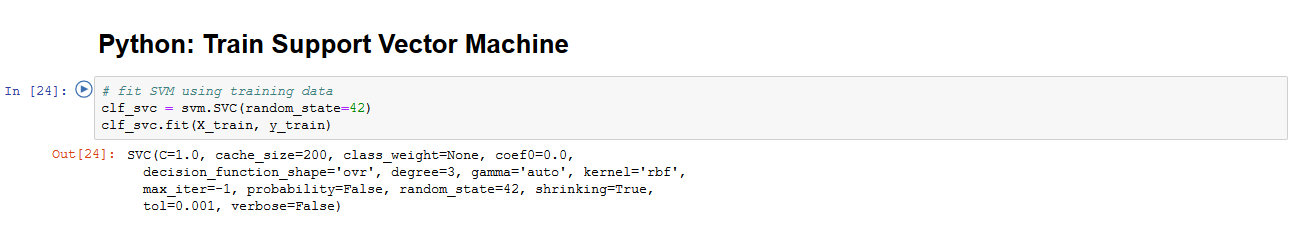
NOTE: if you are familiar with OO, you could see that this could be defined as a single function – **POLYMORPHISM**



Didn’t need scaling function after all – can do it in one line – example of Pythonic approach

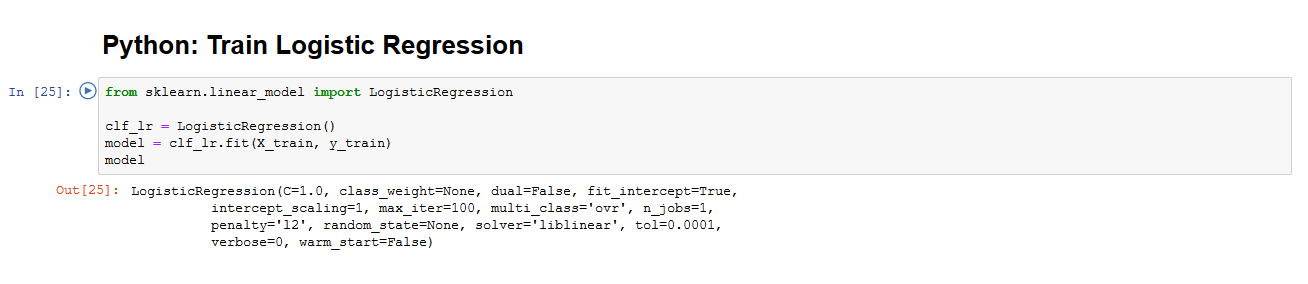


Code to split the dataset into test and train – note that also splits X and Y (X are the columns we train the model with, Y is the target -> churn

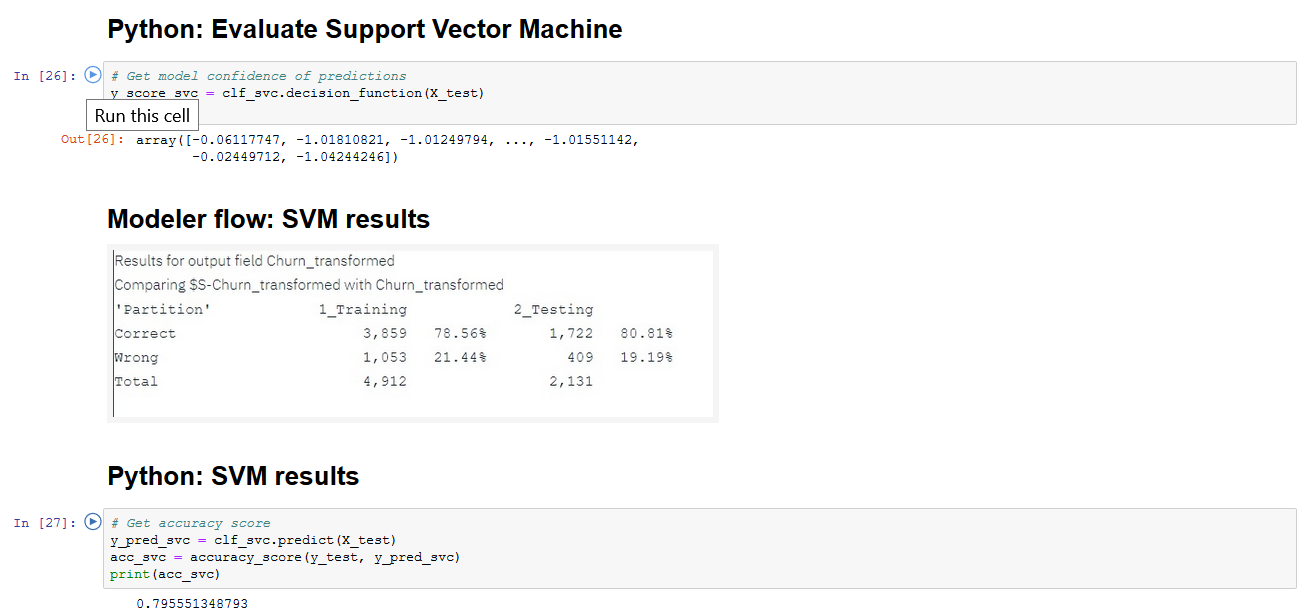


All this code to get to the point where we can train one of the models:

* First line defines the model
* Second line trains the model (also called “fit” the model w. the data)
* Support Vector Classification = SVC – one of the estimators available in SVM
* The **[sklearn.svm](https://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.svm" \o "sklearn.svm)** module includes Support Vector Machine algorithms



Logistic regression train – note that you are passing the same parameters



First cell – run the trained model with the test data as input – this is data that the model has never seen before- get the predictions.

The decision function tells us on which side of the hyperplane generated by the classifier we are (and how far we are away from it). Based on that information, the estimator then label the examples with the corresponding label. FIRST CELL SHOWS WHICH SIDE OF BOUNDARY SVM PUTS INPUT ON, PREDICT GIVES 0 OR 1

