B:A/M! BELGIAN ASSOCIATION OF MARKETING

DISCOVER DATA SCIENCE
ESSENTIALS OF CODING & MODELLING

SAMUEL BORMS

INTRODUCTION

WHO AM I

Sam(uel) Borms

- Data Scientist at Python Predictions since September 2020
- PhD in data science applications of textual sentiment analysis, degree in Business Engineering
- Few months as a consultant in banking (risk management), various internship experiences in finance
- Loves practical coding, mainly in R and Python
- Spent my youth playing soccer, now more into reading and music

WHO ARE YOU?

WHAT IS YOUR EXPERIENCE WITH PROGRAMMING?

PLAN OF ATTACK

OBJECTIVES

This session has following main objectives:

- Introduce you to the data science programming ecosystem
- Introduce you to programming concepts using Python
- Introduce you to the predictive modelling library Cobra and how it connects to the various steps in a typical data science process
- Have some fun with running and modifying actual code
- Point you towards further helpful resources for data science coding

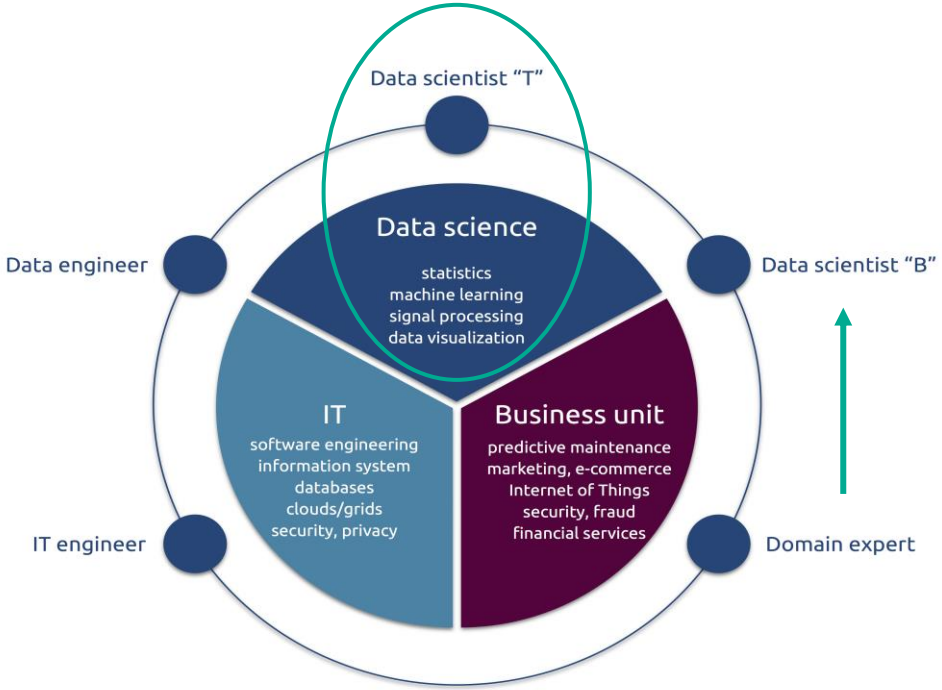
STRUCTURE OF THIS SESSION

Indicative session timing

- 09:30 – 09:45 | introduction, set-up
- 09:45 – 10:00 | data science programming ecosystem
- 10:00 – 10:30 | programming with Python
- 10:30 – 10:40 | **break**
- 10:40 – 11:00 | programming with Python – *HANDS-ON*
- 11:00 – 11:10 | predictive modelling (recap)
- 11:10 – 11:40 | predictive modelling with Cobra
- 11:40 – 11:50 | **break**
- 11:50 – 12:30 | predictive modelling with Cobra – *HANDS-ON*
- 12:30 – 12:40 | closing, further resources

“KNOW WHAT IS POSSIBLE, MASTER WHAT IS NEEDED”

The focus of today is on the role of the Data Scientist and its analytics programming toolkit, but the goal is not to make you one yourself



THE DATA SCIENCE PROGRAMMING ECOSYSTEM

INFORMATION OVERLOAD AHEAD

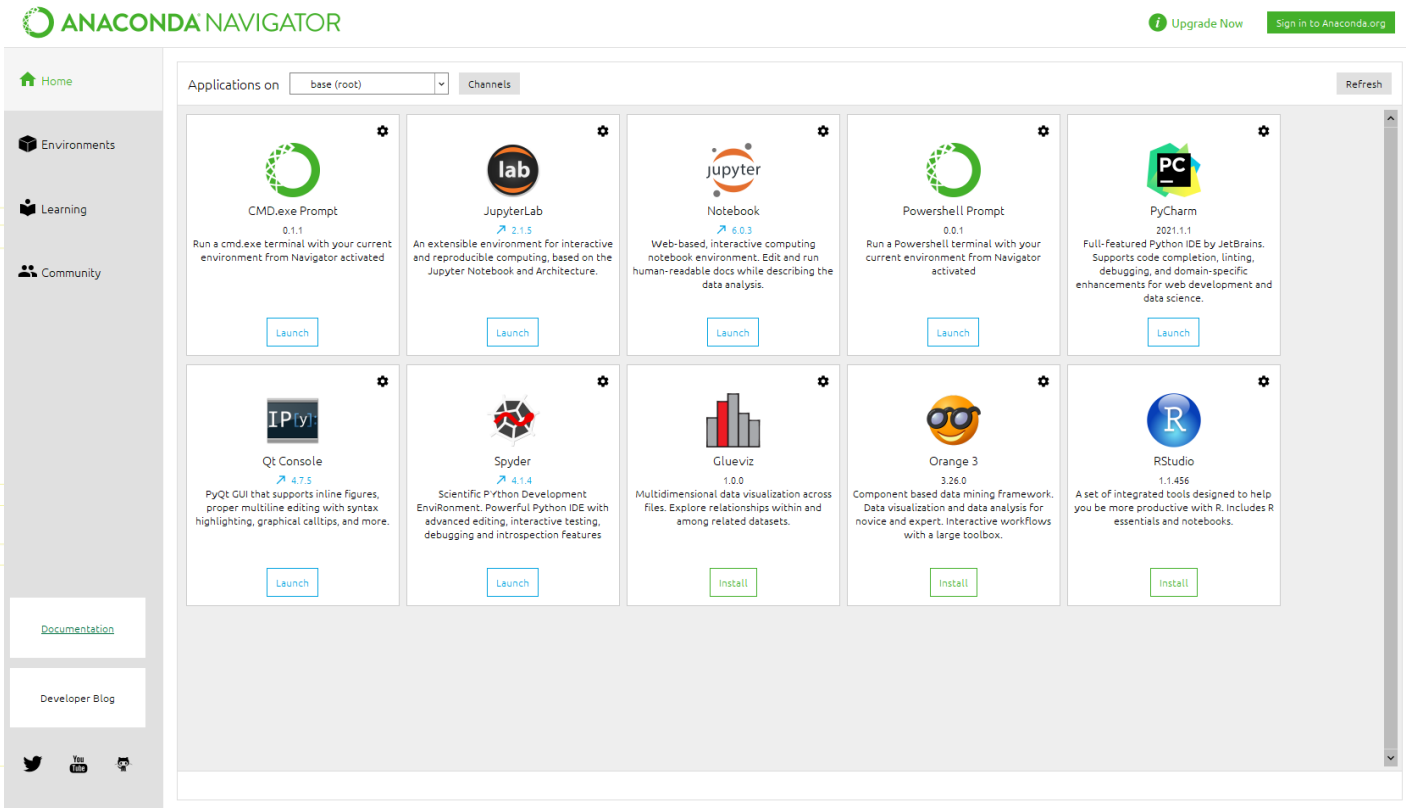
WHAT I HAVE

This is basically what I use for my day-to-day job as a data scientist

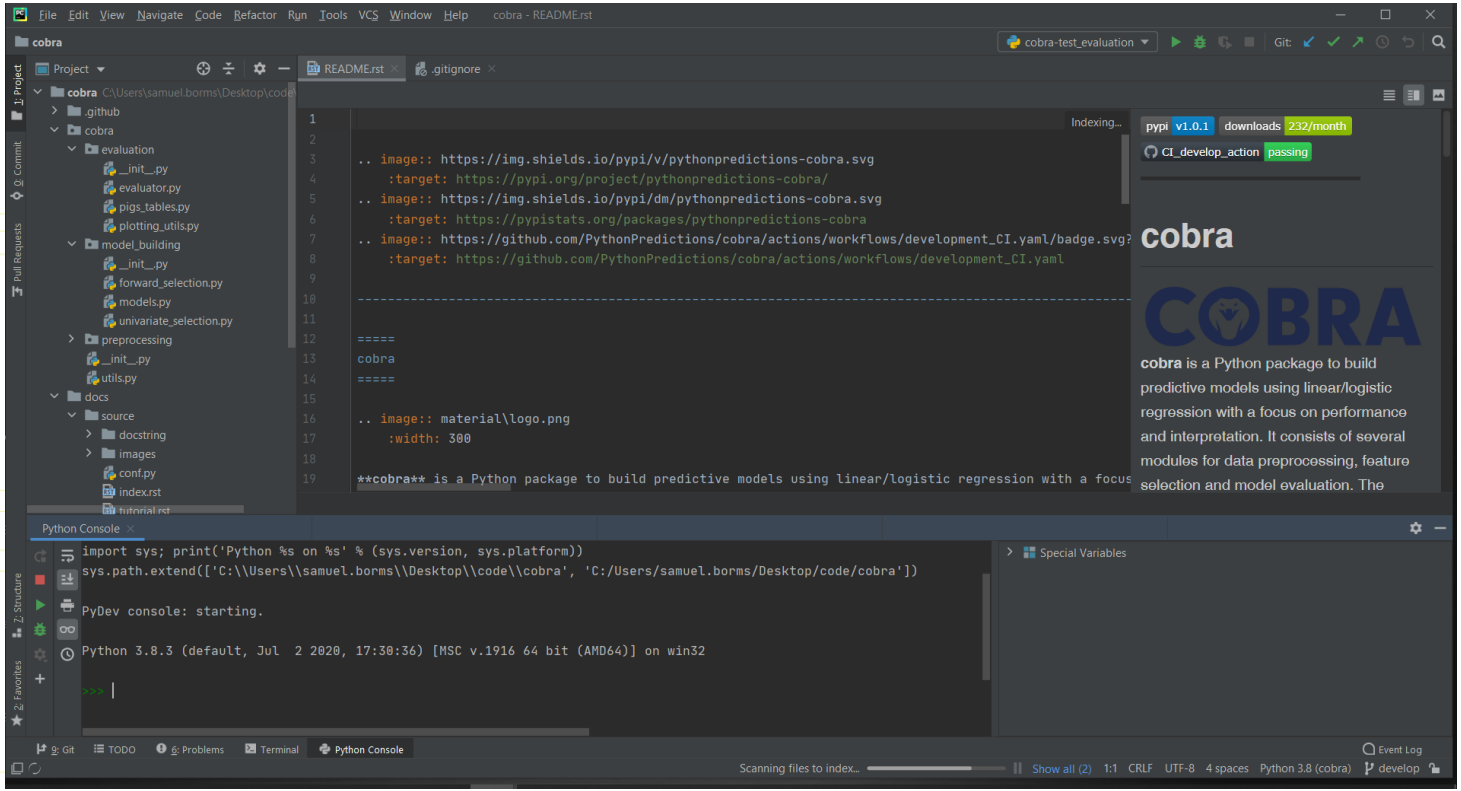
ANACONDA

One-click download (sort of) that includes Python and the most recent and important data science packages, and useful code editors

ANACONDA


ANACONDA NAVIGATOR
Upgrade Now
Sign in to Anaconda.org
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
base (root)
Channels
Refresh
CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch
JupyterLab
2.1.5
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
6.0.3
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated
Launch
PyCharm
2021.1.1
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific enhancements for web development and data science.
Launch
Qt Console
4.7.5
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
Spyder
4.1.4
Scientific Python Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
Glueviz
1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
Orange 3
3.26.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install

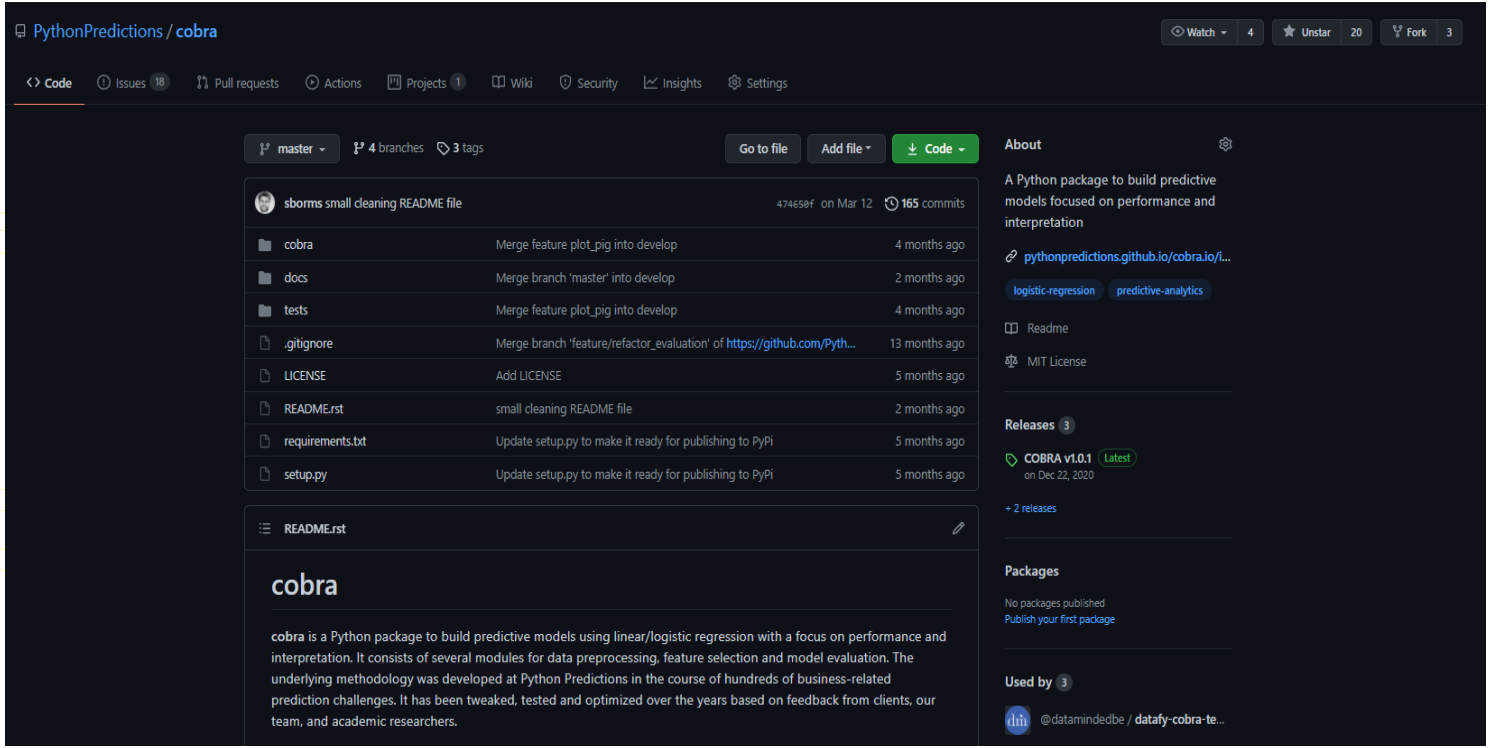
PYCHARM


File Edit View Navigate Code Refactor Run Tools VCS Window Help
cobra - README.rst
cobra
cobra-test_evaluation
Git:
Project
README.rst
.gitignore
cobra C:\Users\samuel.borms\Desktop\code
.github
cobra
evaluation
__init__.py
evaluator.py
pigs_tables.py
plotting_utils.py
model_building
__init__.py
forward_selection.py
models.py
univariate_selection.py
preprocessing
__init__.py
utils.py
docs
source
docstring
images
conf.py
index.rst
tutorial.rst
Indexing...
.. image:: https://img.shields.io/pypi/v/pythonpredictions-cobra.svg
:target: https://pypi.org/project/pythonpredictions-cobra/
.. image:: https://img.shields.io/pypi/dm/pythonpredictions-cobra.svg
:target: https://pypistats.org/packages/pythonpredictions-cobra
.. image:: https://github.com/PythonPredictions/cobra/actions/workflows/development_CI.yaml/badge.svg?
:target: https://github.com/PythonPredictions/cobra/actions/workflows/development_CI.yaml
=====
cobra
=====
.. image:: material\logo.png
:width: 300
cobra is a Python package to build predictive models using linear/logistic regression with a focus
pypi v1.0.1 downloads 232/month
CI_develop_action passing
cobra
COBRA
cobra is a Python package to build predictive models using linear/logistic regression with a focus on performance and interpretation. It consists of several modules for data preprocessing, feature selection and model evaluation. The
Python Console
import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\samuel.borms\\Desktop\\code\\cobra', 'C:/Users/samuel.borms/Desktop/code/cobra'])
PyDev console: starting.
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] on win32
>>>
Special Variables
9: Git
TODO
6: Problems
Terminal
Python Console
Event Log
Scanning files to index...
Show all (2)
1:1
CRLF
UTF-8
4 spaces
Python 3.8 (cobra)
develop

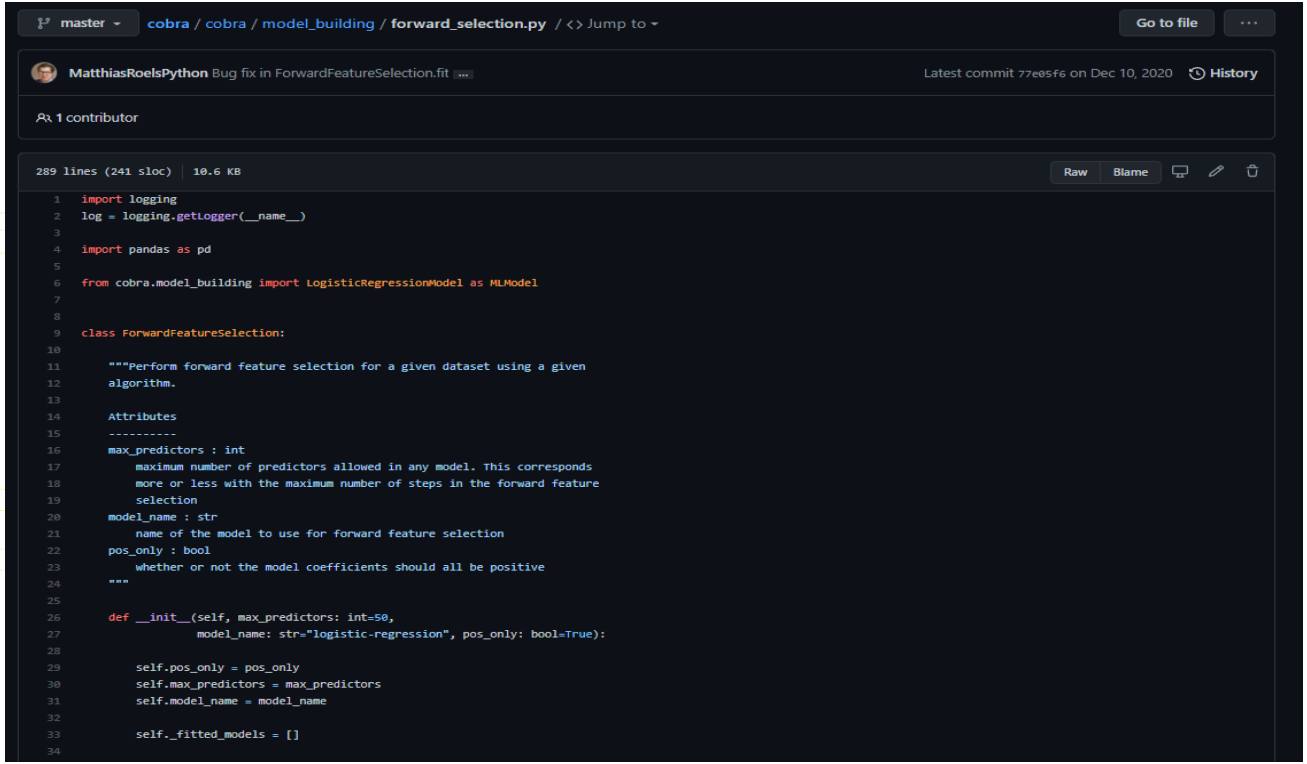
GIT

Git and GitHub (and GitLab, amongst others) house open-source code repositories (private & public), and allow to easily collaborate on projects

GITHUB


PythonPredictions / cobra
Watch 4
Unstar 20
Fork 3
Code
Issues 18
Pull requests
Actions
Projects 1
Wiki
Security
Insights
Settings
master
4 branches
3 tags
Go to file
Add file
Code
sborms small cleaning README file
47465ef on Mar 12
165 commits
cobra Merge feature plot_pig into develop 4 months ago
docs Merge branch 'master' into develop 2 months ago
tests Merge feature plot_pig into develop 4 months ago
.gitignore Merge branch 'feature/refactor_evaluation' of https://github.com/Pyth... 13 months ago
LICENSE Add LICENSE 5 months ago
README.rst small cleaning README file 2 months ago
requirements.txt Update setup.py to make it ready for publishing to PyPi 5 months ago
setup.py Update setup.py to make it ready for publishing to PyPi 5 months ago
README.rst
cobra
cobra is a Python package to build predictive models using linear/logistic regression with a focus on performance and interpretation. It consists of several modules for data preprocessing, feature selection and model evaluation. The underlying methodology was developed at Python Predictions in the course of hundreds of business-related prediction challenges. It has been tweaked, tested and optimized over the years based on feedback from clients, our team, and academic researchers.
About
A Python package to build predictive models focused on performance and interpretation
pythonpredictions.github.io/cobra.io/i...
logistic-regression
predictive-analytics
Readme
MIT License
Releases 3
COBRA v1.0.1 Latest
on Dec 22, 2020
+ 2 releases
Packages
No packages published
Publish your first package
Used by 3
@datamindedbe / datafy-cobra-te...

GITHUB



master cobra / cobra / model_building / forward_selection.py / Jump to Go to file ...

MatthiasRoelsPython Bug fix in ForwardFeatureSelection.fit ... Latest commit 77e05f6 on Dec 10, 2020 History

1 contributor

289 lines (241 sloc) | 10.6 KB Raw Blame

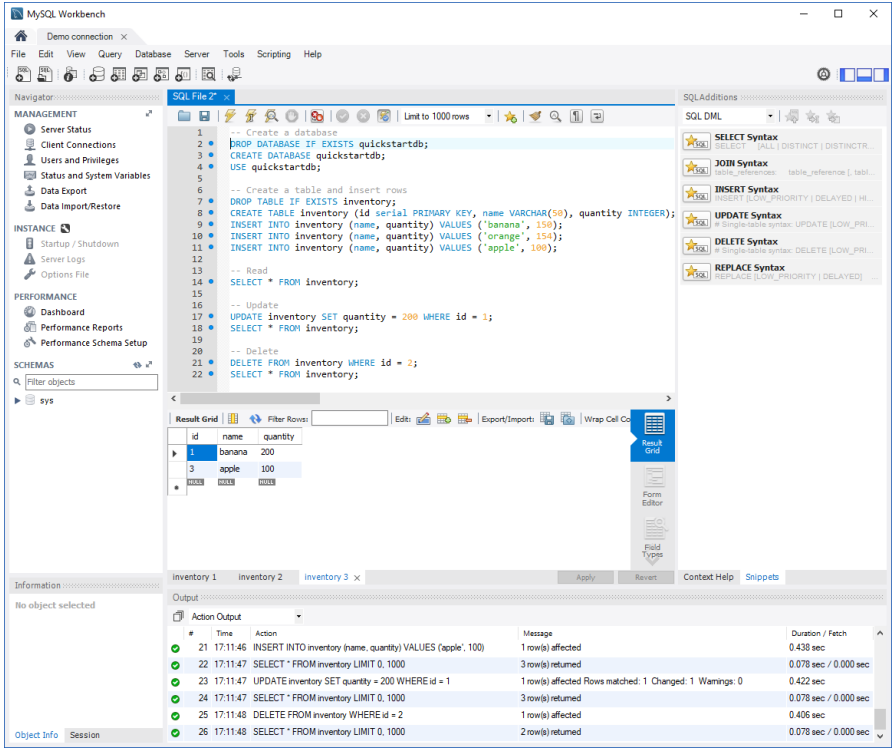
```python
import logging
log = logging.getLogger(__name__)

import pandas as pd

from cobra.model_building import LogisticRegressionModel as MLModel


class ForwardFeatureSelection:

    """Perform forward feature selection for a given dataset using a given
    algorithm.

    Attributes
    ----------
    max_predictors : int
        maximum number of predictors allowed in any model. This corresponds
        more or less with the maximum number of steps in the forward feature
        selection
    model_name : str
        name of the model to use for forward feature selection
    pos_only : bool
        whether or not the model coefficients should all be positive
    """

    def __init__(self, max_predictors: int=50,
                 model_name: str="logistic-regression", pos_only: bool=True):

        self.pos_only = pos_only
        self.max_predictors = max_predictors
        self.model_name = model_name

        self._fitted_models = []

```

TERMINAL

Once in a while you need to work in a terminal – which is less scary than it seems

SQL

Database management and interaction



THERE IS A LOT MORE TO EXPLORE...

A lot of data science is about learning new tools and techniques on-the-fly

- Blogs:
  - https://towardsdatascience.com
  - https://medium.com/topic/data-science
- Community:
  - https://www.kaggle.com
  - https://stackoverflow.com
- Courses:
  - https://learn.datacamp.com
  - Coursera, edX, Udemy

PROGRAMMING WITH PYTHON

COVERAGE

What we will discuss

- Libraries
- Variables & data structures
- For loops & conditional statements
- Functions & parameters
- Rectangular data & filtering
- Object-oriented programming (OOP)

COVERAGE

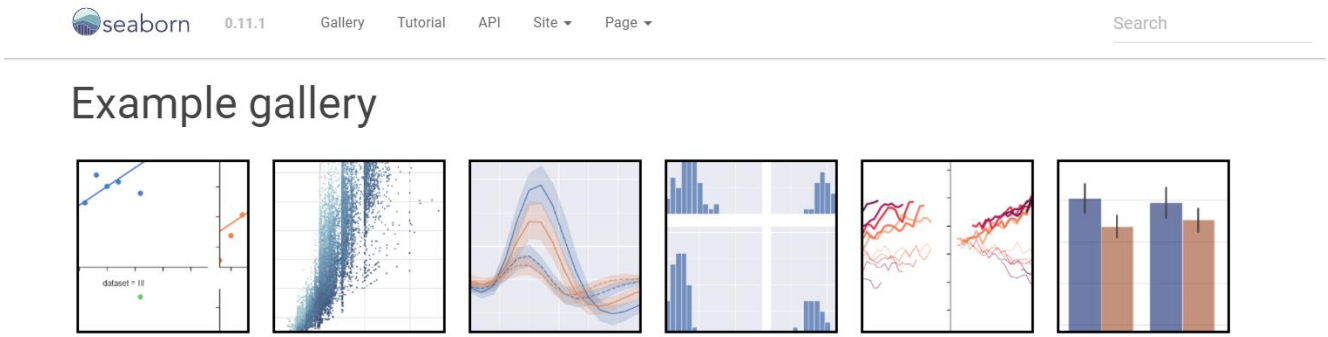
What we will discuss

- **Libraries**
- Variables & data structures
- For loops & conditional statements
- Functions & parameters
- Rectangular data & filtering
- Object-oriented programming (OOP)

WHAT IS A LIBRARY?

A library (or a package) is a collection of various modules and submodules, with Python functions you can use for specific functionality

- No package, no party
  - You load all the useful modules from various packages at the start of your script
- True power of open-source software: you rarely start from scratch, as surely someone has already done what you want to do



DOWNLOAD VIA PIP OR CONDA

The majority of packages sit in the Python Package Index (PyPi) repository

- To install a package from PyPi, download pip (if not available already), and do “pip install …”
- If you installed Anaconda, you can also do “conda install …”

KEY DATA SCIENCE LIBRARIES

These libraries are typically what you need for any data science project

- NumPy for **scientific computing**
- Pandas for **data wrangling**
- scikit-learn for **machine learning**
  - Cobra ☺
- Matplotlib and Seaborn for **visualization**
- (A fancy deep learning library)
- … Most are shipped with Anaconda, so no download struggle

ACCESSING A FUNCTION FROM A PACKAGE

Three ways to load a function using the import keyword

```
from numpy.random import rand # function straight from submodule
rand(5) # five random numbers between 0 and 1
```

```
array([0.23815297, 0.170851  , 0.87182124, 0.72535266, 0.75651519])
```

```
import numpy.random as npr # load submodule
npr.rand(5)
```

```
array([0.300613  , 0.78355676, 0.21172528, 0.84555395, 0.7120013 ])
```

```
import numpy as np # load main NumPy module
np.random.rand(5)
```

```
array([0.2936398 , 0.14178876, 0.66282765, 0.31799115, 0.42896267])
```

COVERAGE

What we will discuss

- Libraries
- **Variables & data structures**
- For loops & conditional statements
- Functions & parameters
- Rectangular data & filtering
- Object-oriented programming (OOP)

WHAT IS A VARIABLE?

Variables are names that can be assigned a value and then used to refer to that value throughout your code

- To keep values accessible, and be able to re-use them
- To give context to values
  - Make sure your collaborator can understand what you are trying to do

VARIABLE ASSIGNMENT

A variable is assigned to a name using the “=” operator

- Be careful, “==” means testing for “is equal to”
- Compared to low-level programming languages, you do not have to specify the data type of any variable (= dynamic typing)
- You can use most names, except reserved keywords (e.g. for, import, in)

```
session = 2
type(session)
```

```
int
```

```
session += 1
print(session)
```

```
3
```

WHAT ARE DATA STRUCTURES?

A data structure is a predefined way in which data can be organized and manipulated

- Two main built-in data structures:
  - Lists
  - Dictionaries
  - … there are others (tuples, sets)
- Not to be confused with data types (= one level lower):
  - Integers
  - Floats
  - Strings
  - Booleans
- You have the control to create your own data structures (cf. OOP)

LISTS

A list is a mutable ordered sequence of items; access through position-based indexing

```
list_of_strategies = ["up-sell", "cross-sell", "down-sell"]
```

```
len(list_of_strategies)
```

```
3
```

```
list_of_strategies[0] # access through indexing (first element is at index 0)
```

```
'up-sell'
```

LISTS

You can interact with lists using its methods (e.g. append, insert, extend)

```
list_of_strategies.append("stay put")
```

```
list_of_strategies
```

```
['up-sell', 'cross-sell', 'down-sell', 'stay put']
```

DICTIONARIES

A dict is a mutable unordered set of key-value pairs; access through the unique keys

```
dict_of_participants = { # collection of key-value pairs
    "Jan": ("XYZ", 42),
    "Sam": ("ABC", 28),
    "Daphne": ("MNO", 35)
}
```

```
dict_of_participants["Sam"] # you can only access the data via a key
```

```
('ABC', 28)
```

This is a tuple, an immutable data structure

COVERAGE

What we will discuss

- Libraries
- Variables & data structures
- **For loops & conditional statements**
- Functions & parameters
- Rectangular data & filtering
- Object-oriented programming (OOP)

WHAT IS A FOR LOOP?

With for loops one can execute a set of statements one-by-one

- Do “something” for “every element in this sequence”
- Use of “break” and “continue” statements to have more control over what happens within the for loop

FOR LOOP

```
for strategy in list_of_strategies:
    print("Possible strategy:", strategy)
```

```
Possible strategy: up-sell
Possible strategy: cross-sell
Possible strategy: down-sell
Possible strategy: stay put
```

WHAT IS A CONDITIONAL STATEMENT?

Conditional statements allow to execute chunks of code if particular conditions are met

- Conceptualized using the “if”-“elif”-“else” structure

- If “this is true”, then do “that”, if not …
- … then if “this other thing is true”, then do “that other thing”, if not …
- … do “a final thing”

CONDITIONAL STATEMENT

```
for participant in dict_of_participants:
    print("name:", participant)
    if participant == "Daphne":
        value = dict_of_participants[participant]
        age = value[1]
        print("   Age:", age)
    elif participant == "Jan":
        value = dict_of_participants[participant]
        company = value[0]
        print("   Company:", company)
    else:
        print("   We do not want to know your info.")
```

```
name: Jan
   Company: XYZ
name: Sam
   We do not want to know your info.
name: Daphne
   Age: 35
```

COVERAGE

What we will discuss

- Libraries
- Variables & data structures
- For loops & conditional statements
- **Functions & parameters**
- Rectangular data & filtering
- Object-oriented programming (OOP)

WHAT IS A FUNCTION?

A function is an encapsulation of (i) taking certain input, (ii) performing a set of manipulations on the input, and (iii) returning a certain output

- Very powerful mechanism to make your code modular and re-usable
  - Difficulty: make it “pure” (= no side effects) with clearly defined in- and outputs
- The input is defined by a set of **parameters**, that changes the output and the type of manipulations done
- Data structures often have particular functions (“methods”) that can be applied to them (cf. append() for lists)

AN EXAMPLE FUNCTION

```
def get_age(participants, who):
    """
    Parameters
    ------------
    - participants : dict
        Dictionary of participants

    - who : str
        Name of participant to get age from, as a string
        Must be a valid key of 'participants' argument

    Returns
    ------------
    Age of selected participant
    """
    value = participants[who]
    age = value[1]

    return age
```

```
get_age(dict_of_participants, "Sam")
```

```
28
```

COVERAGE

What we will discuss

- Libraries
- Variables & data structures
- For loops & conditional statements
- Functions & parameters
- **Rectangular data & filtering**
- Object-oriented programming (OOP)

WHAT IS RECTANGULAR DATA?

Most data in a data science context is structured in rectangular format (i.e. as a matrix)

- Rows: observations
  - customer, time
- Columns: variables
  - age, gender, GDP, sales
- Cells: values
  - 16, female, 3%, 150000

	Artist	Genre	Listeners	Plays
0	Billie Holiday	Jazz	1,300,000	27,000,000
1	Jimi Hendrix	Rock	2,700,000	70,000,000
2	Miles Davis	Jazz	1,500,000	48,000,000
3	SIA	Pop	2,000,000	74,000,000

THE PANDAS LIBRARY TO DEAL WITH RECTANGULAR DATA

Rectangular data is most efficiently processed with the Pandas library – the matrices are called DataFrames

- Huge built-in functionality to analyse your data
  - Summary statistics (minimum, maximum, average, quantile)
  - Grouping
  - Plotting
  - Merging various DataFrames
  - File input & output
  - Derive extra columns
  - Filtering

- **Filtering** is obtaining a subset of the data you are interested in, often based on a condition (e.g. take observations where age > 65)

A PANDAS DATAFRAME

```
import pandas as pd
```

```
data = {
    "Artist": ["Billy Holiday", "Jimi Hendrix", "Miles Davis", "SIA"],
    "Genre": ["Jazz", "Rock", "Jazz", "Pop"],
    "Listeners": [1300000, 2700000, 1500000, 2000000],
    "Plays": [27000000, 70000000, 48000000, 74000000]
}

df = pd.DataFrame(data)

df
```

	Artist	Genre	Listeners	Plays
0	Billy Holiday	Jazz	1300000	27000000
1	Jimi Hendrix	Rock	2700000	70000000
2	Miles Davis	Jazz	1500000	48000000
3	SIA	Pop	2000000	74000000

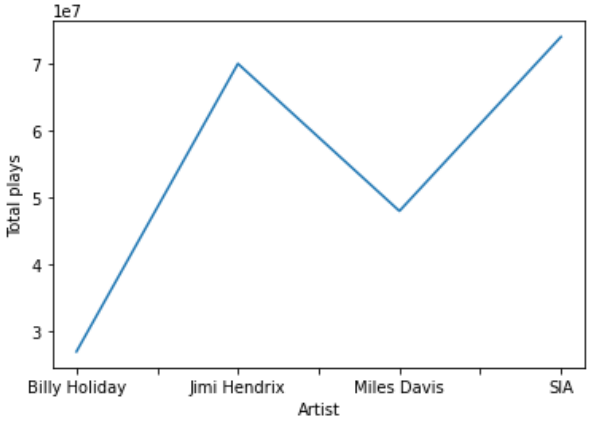
PANDAS BUILT-IN ANALYSIS FUNCTIONS

```
df.mean()
```

```
Listeners     1875000.0
Plays        54750000.0
dtype: float64
```

```
df.set_index("Artist")["Plays"].plot(ylabel="Total plays")
```

```
<AxesSubplot:xlabel='Artist', ylabel='Total plays'>
```



FILTERING A DATAFRAME

```
df_jazz = df[df["Genre"] == "Jazz"]

df_jazz
```

	Artist	Genre	Listeners	Plays
0	Billy Holiday	Jazz	1300000	27000000
2	Miles Davis	Jazz	1500000	48000000

```
df_popular = df[df["Listeners"] >= 2000000]

df_popular
```

	Artist	Genre	Listeners	Plays
1	Jimi Hendrix	Rock	2700000	70000000
3	SIA	Pop	2000000	74000000

COVERAGE

What we will discuss

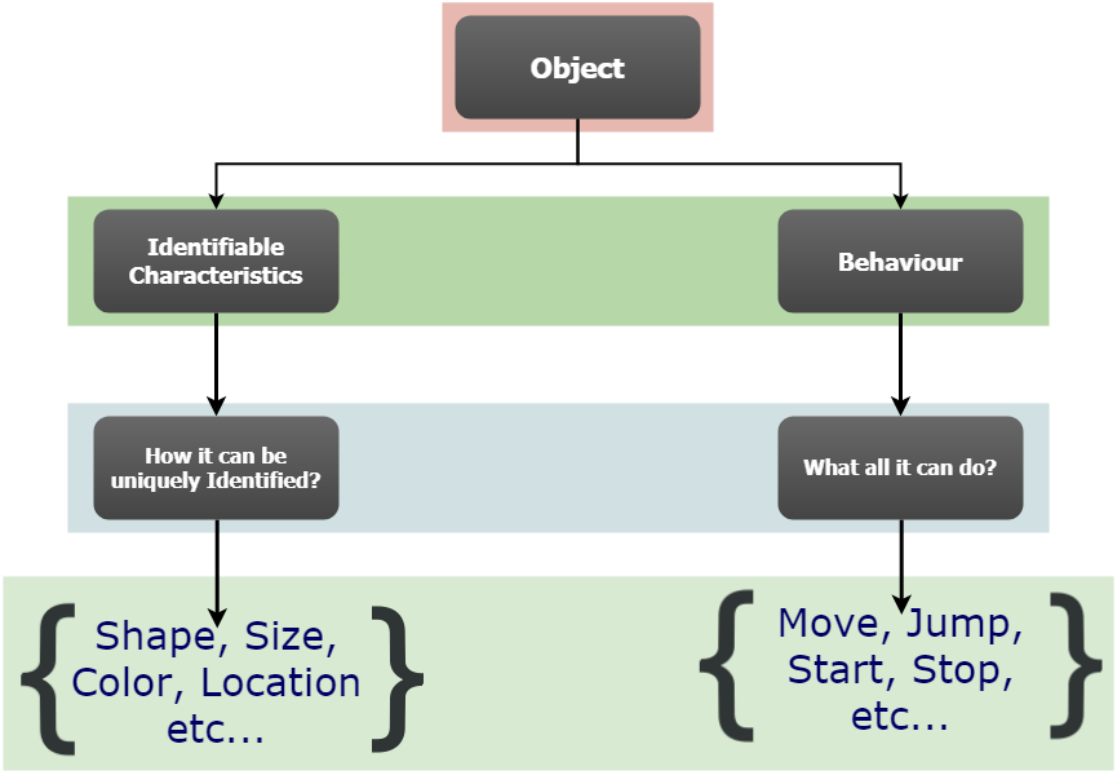
- Libraries
- Variables & data structures
- For loops & conditional statements
- Functions & parameters
- Rectangular data & filtering
- **Object-oriented programming (OOP)**

WHAT IS OBJECT-ORIENTED PROGRAMMING?

OOP is an approach to structuring your program by bundling related characteristics and behaviours into individual objects

- Characteristics = variables with data, behaviours = functions (methods)
- A “class” is a blueprint of an object
- Again, allows to abstract away unnecessary complexity and repetition
- Don’t worry, Python is also a very **functional programming** language in many respects, especially when it comes to data science – OOP is mainly for developers, yet you need to understand it so you can benefit from it

WHAT IS OBJECT-ORIENTED PROGRAMMING?



A CLASS

Setting up an object class in Python is easy, you at minimum require the __init__(self, ...) initialization function

```
class Rectangle:
    def __init__(self, length, breadth, unit_cost=0):
        self.length = length
        self.breadth = breadth
        self.unit_cost = unit_cost
    def get_perimeter(self):
        return 2 * (self.length + self.breadth)
    def get_area(self):
        return self.length * self.breadth
    def calculate_cost(self):
        area = self.get_area()
        return area * self.unit_cost
```

INTERACTING WITH YOUR CREATED OBJECT

You can interact with the actual object you initialized from your class, using the functions you defined within the class

```
r = Rectangle(160, 120, unit_cost=2000)

print("Area of rectangle: %s cm^2" % (r.get_area()))
print("Cost of rectangular field: EUR%s " %(r.calculate_cost()))
```

```
Area of rectangle: 19200 cm^2
Cost of rectangular field: EUR38400000
```

PRACTICE TIME – 20’

Fool around with your personal notebook in Kaggle (part A. Programming with Python)

https://www.kaggle.com/sborms/cobra-tutorial-bam-06052021-main

- The best way to learn is to mess up and understand why
- You can try one of the three exercises at the end
- Ask me if you are stuck!

PREDICTIVE MODELLING...

WHO REMEMBERS THE FIVE STEPS IN PREDICTIVE MODELLING?

Project Definition

Data Preparation

Model Building

Model Validation

Model Usage

Project Definition

take order

check stock

check timing

understand what to predict

check data availability

create project plan

Data Preparation

gather vegetables	collect data
clean vegetables	clean data
cut vegetables	apply data transformations

Model Building

choose ingredients	select variables
find ingredient proportions	find variable weights
choose cooking technique	choose algorithm

Model Validation

taste soup	validate model
approve soup	select champion model

Model Usage

arrange on plate	profile results
serve dish	present results
‘everything fine?’	execute field test

... WITH COBRA

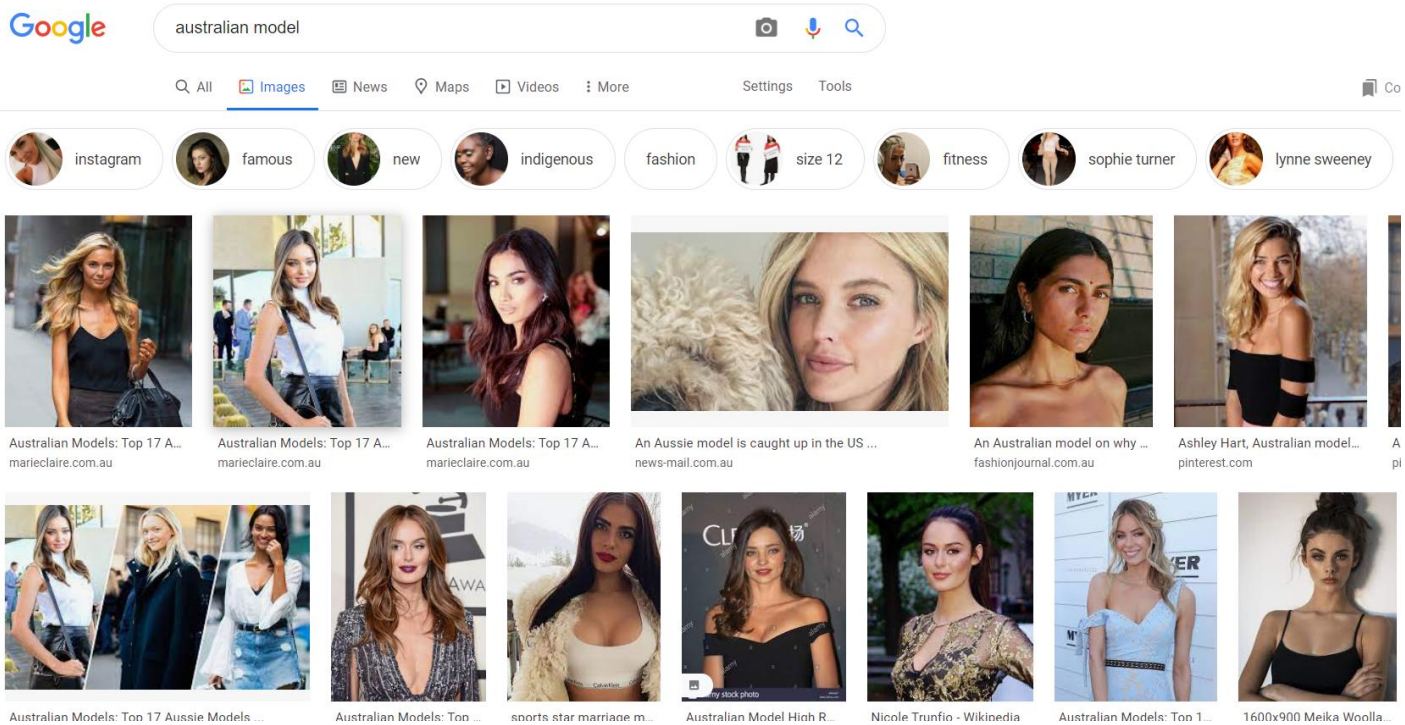
WHAT IS COBRA?

Cobra is an open-source Python library to build predictive models

- Quick
- Interpretable
- High predictive power
- Rooted in industry expertise (built by Python Predictions)

COBRA

SOME MODEL BASICS FIRST



LOGISTIC REGRESSION

$$p = \frac{1}{1+e^{-(\alpha + \beta * X)}}$$

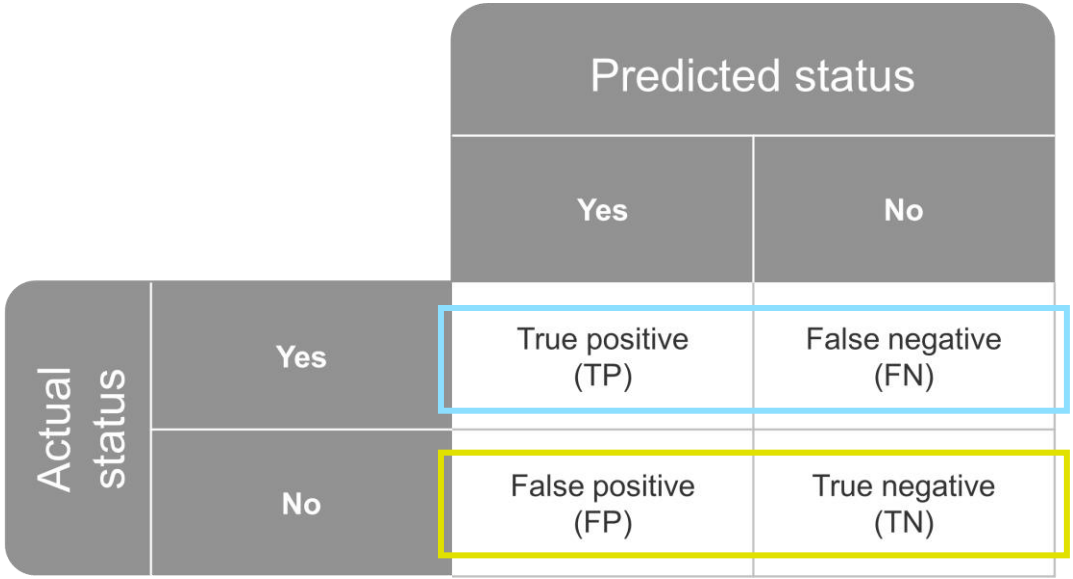
probability (0-1)

predictors

Prediction of the **target** based on a cut-off (e.g. > 0.50 is positive)

$$p = \frac{1}{1+e^{-(\alpha + \beta_1 * X_1 + \beta_2 * X_2 + \ldots)}}$$

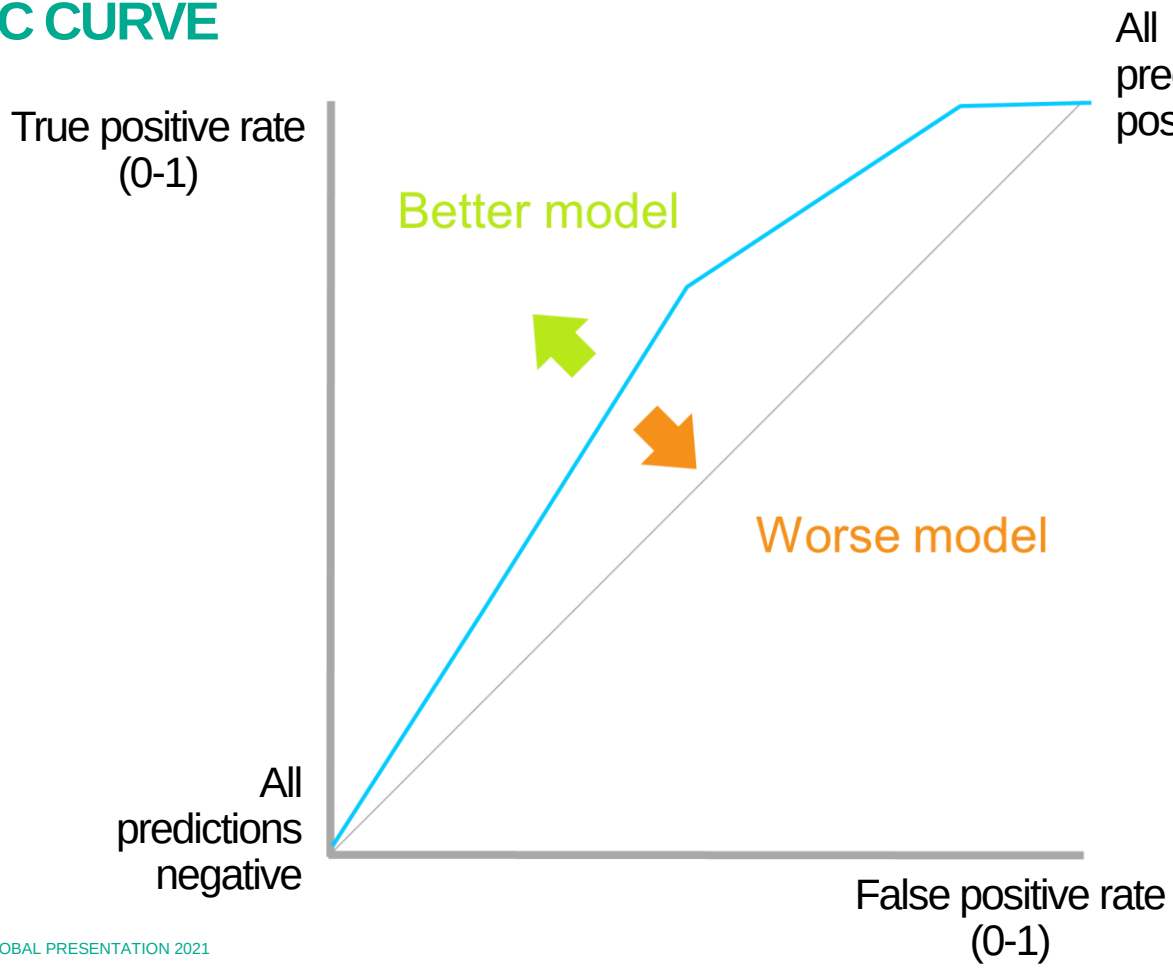
CONFUSION MATRIX



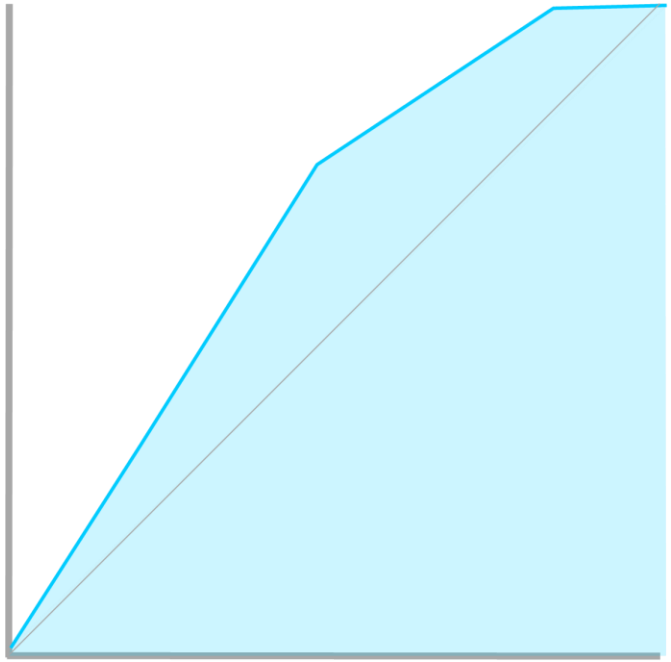
True positive rate (Recall) $= \frac{TP}{TP + FN}$

False positive rate $= \frac{FP}{FP + TN}$

ROC CURVE


All predictions positive
True positive rate (0-1)
Better model
Worse model
All predictions negative
False positive rate (0-1)

AUC



Area under ROC Curve (AUC)

=

1 number that summarizes all combinations

From 0.70, your model can be called decent

PROJECT DEFINITION

Predict whether income exceeds $50k/year based on U.S. census data

- Dataset:
  - Survey of adults and their earnings
  - 14 variables (predictors)
  - Target variable:
    - 1 = income > 50k USD
    - 0 = income <= 50k USD

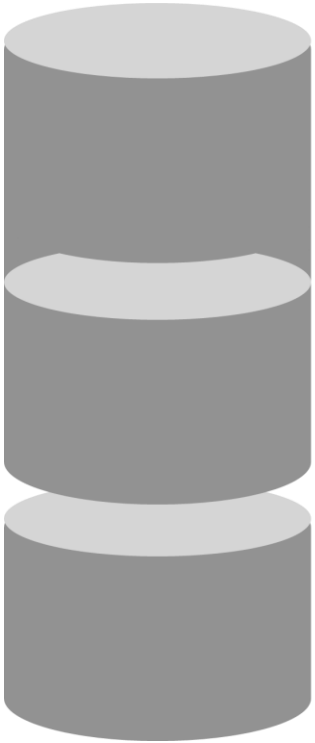
Source: https://archive.ics.uci.edu/ml/datasets/Adultx

DATA PREPARATION

There are a set of <u>preprocessing</u> steps to perform

- Cleaned file to start from: earnings_dataset.csv
- Split into train, selection, and validation sets to avoid overfitting and ensure correct validation
- **Discretization** of continuous variables and **incidence replacement**
- Leads to **analytical basetable (ABT)**

SPLITTING OF YOUR DATA



Training Set

Used to train the model

Selection Set

Used to select the best model

Validation Set

Used to evaluate the model

DISCRETIZATION AND INCIDENCE REPLACEMENT

Discretization and incidence replacement lies at the heart of Cobra

- **Discretization** = convert the continuous variables in groups
  - Example: age in bins of 15-25, 26-35, 36-45, etc.
  - For discrete variables, this is already the case, but some are regrouped
- **Incidence replacement** = replace the bins by their average incidence
  - Example: replace all observations for variable age in the bin 26-35 with 34% (which is, fictively, how many people in that age group earn over 50k USD)
- Advantages
  - Coherent treatment and interpretation of variables
  - Deals with outliers, missing values, and categorical variables

ANALYTICAL BASETABLE

	ID	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	TARGET	split
0	1935	37	Private	193106	Bachelors	13	Never-married	Sales	Not-in-family	White	Female	0	0	30	United-States	0	train
1	18592	56	Self-emp-inc	216636	12th	8	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	1651	40	United-States	0	train
2	12563	53	Private	126977	HS-grad	9	Separated	Craft-repair	Not-in-family	White	Male	0	0	35	United-States	0	train
3	553	72	Private	205343	11th	7	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	0	train
4	3480	46	State-gov	106705	Masters	14	Never-married	Exec-managerial	Not-in-family	White	Female	0	0	38	United-States	0	train

ANALYTICAL BASETABLE (PROCESSED)

workclass_enc	fnlwgt_enc	education_enc	marital-status_enc	occupation_enc	relationship_enc	race_enc	sex_enc	native-country_enc	age_enc	education-num_enc	hours-per-week_enc
0.217162	0.235991	0.413544	0.045656	0.271924	0.102276	0.253332	0.108657	0.244963	0.286853	0.387529	0.093495
0.548056	0.235991	0.071429	0.444599	0.472512	0.446909	0.253332	0.303668	0.244963	0.371879	0.154644	0.210125
0.217162	0.235991	0.158162	0.060016	0.221666	0.102276	0.253332	0.303668	0.244963	0.371879	0.154644	0.093495
0.217162	0.235991	0.050310	0.086460	0.136811	0.059946	0.253332	0.108657	0.244963	0.256250	0.053155	0.210125
0.267214	0.235991	0.550446	0.045656	0.472512	0.102276	0.253332	0.108657	0.244963	0.400843	0.618506	0.210125

GENERAL CODE STRUCTURE

Create instance of PreProcessor object

```
preprocessor = PreProcessor.from_params(parameters)
```

Split data into train-selection-validation sets

```
basetable = preprocessor.train_selection_validation_split(data)
```

Fit the pipeline

```
basetable = preprocessor.fit(basetable)
```

Transform the data

```
basetable = preprocessor.transform(basetable)
```

MODEL BUILDING

The first step is <u>feature preselection</u>

- Check the variables independently and rule out those with little predictive power or prone to overfitting
- Keep all variables of which the single-variable model's **AUC** is above a certain threshold
- You can also inspect and drop the variables with high correlation

GENERAL CODE STRUCTURE

Run univariate preselection procedure and plot output

```
df_auc = univariate_selection.compute_univariate_preselection(basetable, thresholds)
```

```
plot_univariate_predictor_quality(df_auc)
```

Get a list of predictors selected by the univariate selection

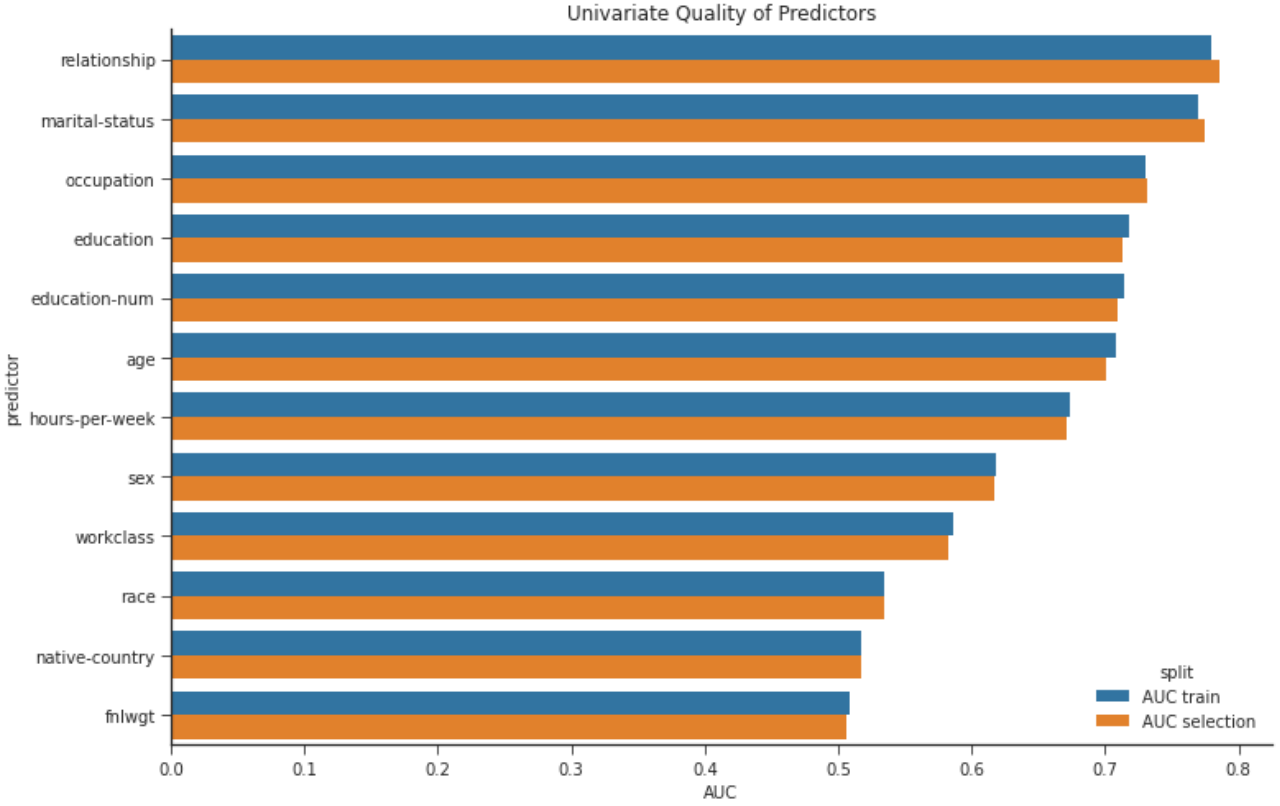
```
preselected_predictors = univariate_selection.get_preselected_predictors(df_auc)
```

Compute and plot correlations between preprocessed predictors

```
df_corr = univariate_selection.compute_correlations(basetable)
```

```
plot_correlation_matrix(df_corr)
```

UNIVARIATE PRESELECTION

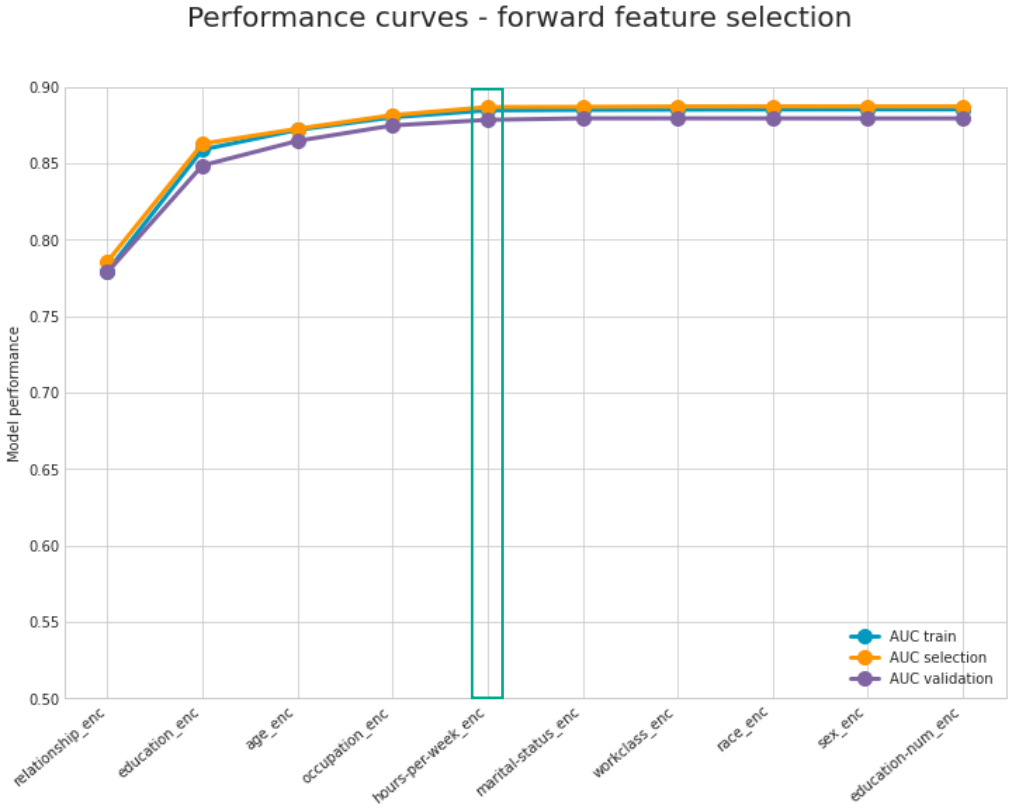


MODEL BUILDING

The second step is <u>forward feature selection</u>

- Add variables sequentially to the model
- Pick the number of variables at the “knack” where the performance flattens

SELECTION OF THE BEST MODEL

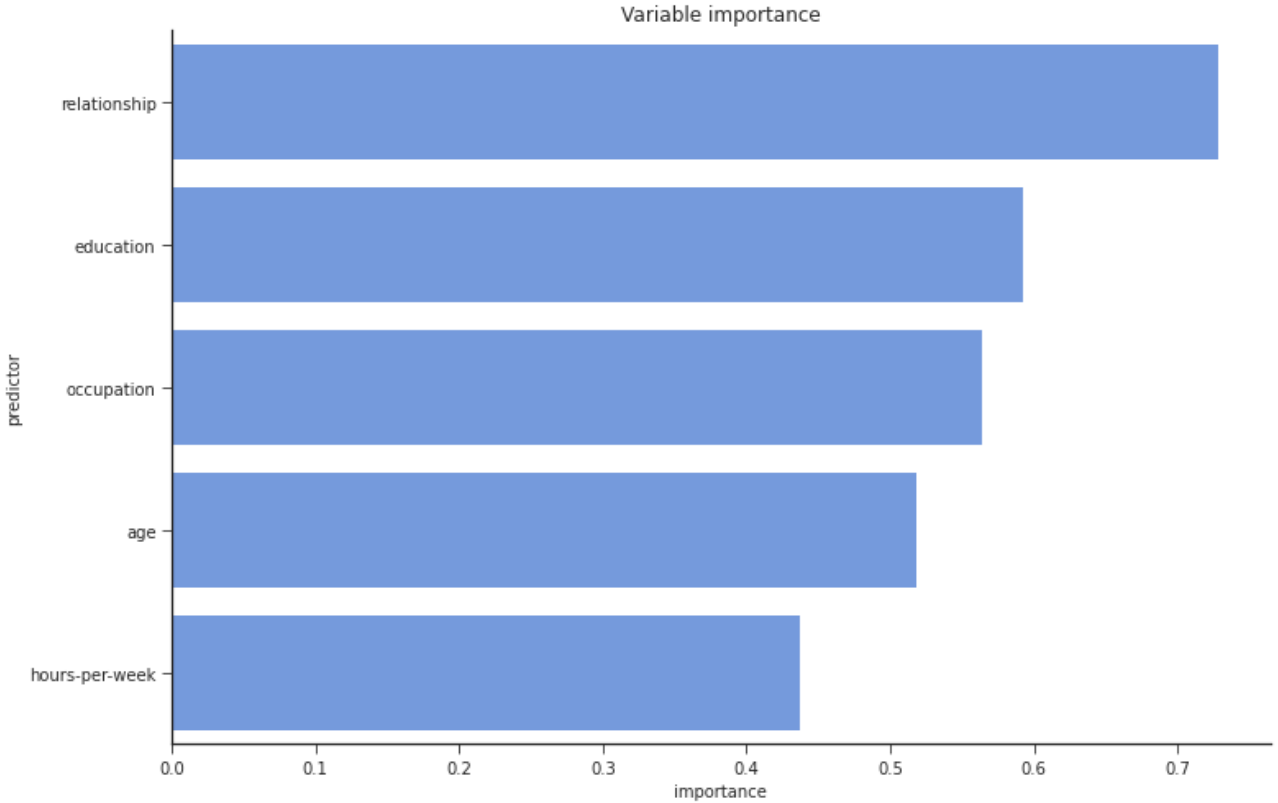


MODEL BUILDING

The final model building step (and in essence first validation step) is analysing the variable importance

- Simple metric: correlation of the variable with the model probability score

VARIABLE IMPORTANCE



GENERAL CODE STRUCTURE

Initialize forward feature selection procedure

```
forward_selection = ForwardFeatureSelection(parameters)
```

```
forward_selection.fit(basetable)
```

Run forward feature selection and plot performance curves

```
performances = forward_selection.compute_model_performances(basetable, target_column_name)
```

```
plot_performance_curves(performances)
```

Select and extract model of choice

```
model = forward_selection.get_model_from_step()
```

```
final_predictors = model.predictors
```

Compute and plot the importance of each predictor in the model

```
variable_importance = model.compute_variable_importance(basetable)
```

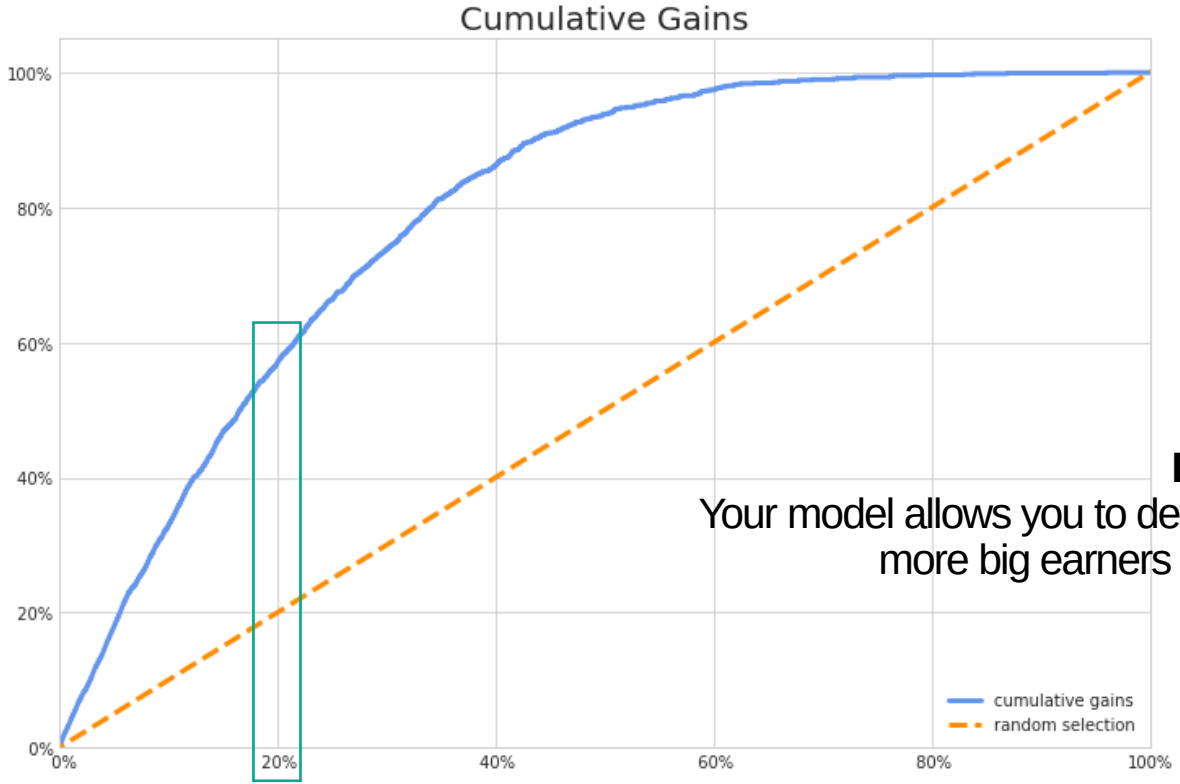
```
plot_variable_importance(variable_importance)
```

MODEL VALIDATION

There are a few common statistics to evaluate how well the model performs on the validation set

- Accuracy, Precision, Recall, F1 (all bundled in the confusion matrix)
- AUC, ROC Curve
- Cumulative Gains Curve, Lift

CUMULATIVE GAINS CURVE AND LIFT



Lift of 3 at 20%

Your model allows you to detect three times more big earners than at random

GENERAL CODE STRUCTURE

Instantiate Evaluator object

```
evaluator = Evaluator()
```

Automatically find the best cut-off probability

```
evaluator.fit()
```

Get and plot various scalar metrics

```
evaluator.scalar_metrics
```

```
evaluator.plot_confusion_matrix()
```

```
evaluator.plot_roc_curve()
```

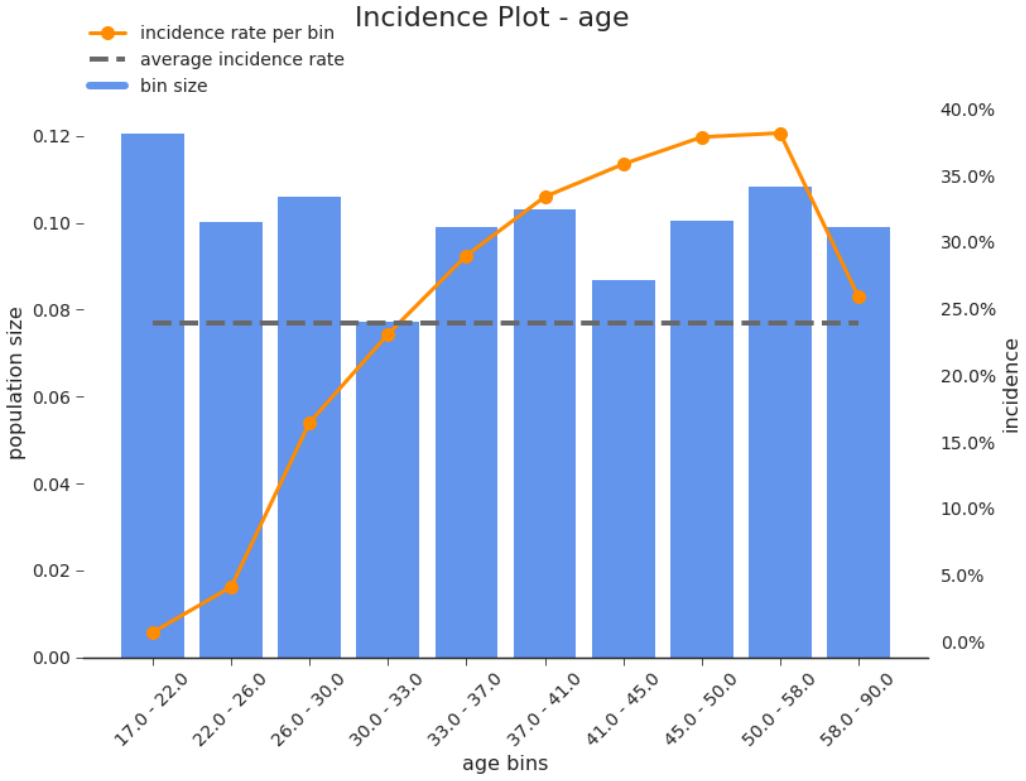
```
...
```

MODEL USAGE

Interpret model in a business context and use it for decision-making

- One useful tool: Predictor Insight Graphs (PIGs)
- To continuously use the model for your decision-making, you will have to **industrialize** (= automate scoring the model and integrate its interpretation) at some point

PIGS FOR BUSINESS



GENERAL CODE STRUCTURE

Generate PIG tables

```
pig_tables = generate_pig_tables(basetable)
```

Plot PIG tables

```
plot_incidence(pig_tables)
```

PRACTICE TIME – 40’

Fool around with your personal notebook in Kaggle (part B. Predictive modelling with Cobra)

https://www.kaggle.com/sborms/cobra-tutorial-bam-06052021-main

- The best way to learn is to mess up and understand why
- Do you understand all the steps?
- Ask me if you are stuck!

CLOSING

HOW DID YOU LIKE IT?

FURTHER RESOURCES

Anaconda (software distribution with key data science Python libraries and editors)

- https://www.anaconda.com/products/individual

Cobra (Python Predictions' predictive modelling library)

- Article: https://www.pythonpredictions.com/news/the-little-trick-we-apply-to-obtain-explainability-by-design
- Code repository: https://github.com/PythonPredictions/cobra
- Documentation: https://pythonpredictions.github.io/cobra.io/index.html
- Example: https://github.com/PythonPredictions/Cobra-DS-meetup-Leuven
- Talk: https://www.youtube.com/watch?v=w7ceZZqMEaA&feature=youtu.be

Kaggle (data science community – competitions, examples and datasets)

- https://www.kaggle.com

FURTHER RESOURCES

Python (starter's material)

- https://realpython.com/python-basics

Logistic regression (and classification in general)

- https://developers.google.com/machine-learning/crash-course/logistic-regression/video-lecture

THANK YOU

CONTACT
Samuel Borms
Python Predictions
sam.borms@pythonpredictions.com

Belgian Association of Marketing vzw/asbl
Z1 Research Park, 120, 1731 Zellik – T. +32 2 234 54 00 – E. info@marketing.be – www.marketing.be