Webinars

Welcome!

The program will begin soon.

You will not hear audio until we begin.

SPE International

Webinars

SPE Webinars are Sponsored by:

SPE

SPE Foundation

SPE International

Webinars

Data Science using Python in Petroleum engineering applications: a kick start in Python

Dr. Luigi Saputelli

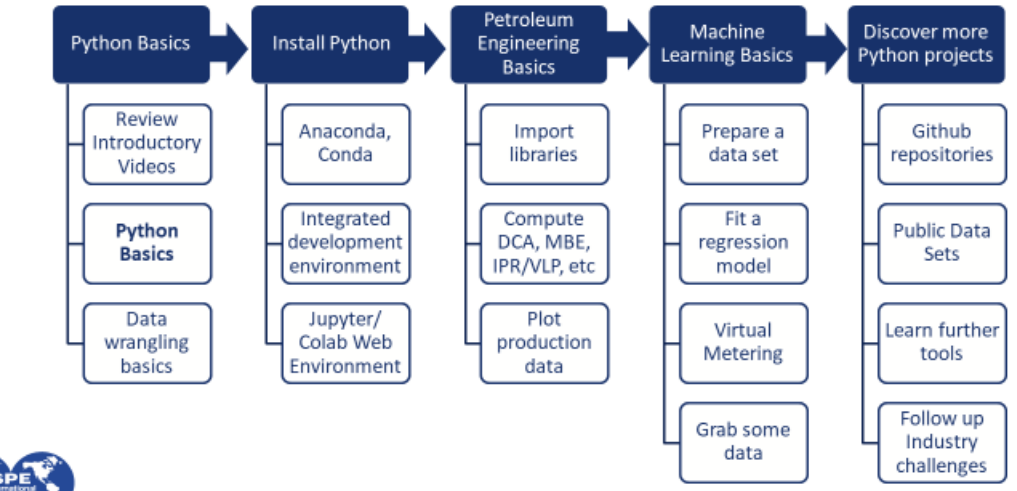
May 19, 2021

SPE International

19 MAY 2021

AGENDA

- Introduction to Statistics, Analytics, AI and Deep learning
- Python Basics
- Python Basics in PE
- Machine Learning Basics
- Virtual Metering Use Case
- Resources
- Way Forward



Dr. Luigi Saputelli

- Reservoir Engineering Expert Advisor with 30 years of experience as reservoir, drilling and production engineer in PDVSA, ADNOC, Hess and Halliburton.
- Researcher, lecturer, and active volunteer in the Society of Petroleum Engineers (SPE) for 25 years, where he served as JPT editor (2012-2020), Production and Operations Advisory Board since 2010, founding member of Real-time Optimization Interest Group and Petroleum Data-driven Analytics technical sections. He is recipient of the 2015 SPE International Production and Operations Award.
- He has published over 100 industry papers on digital oilfield, reservoir management and real-time production optimization.
- BSc in Electronic Engineer from Universidad Simon Bolivar (1990), with a MSc in Petroleum Engineering from Imperial College (1996), and a PhD in chemical engineering from University of Houston (2003).
- He is also serving as managing partner in Frontender, a services firm in Houston. He can be reached at LSaputelli@frontender.com

Why Data Science?

Why Data Science in Petroleum Engineering?

- Better informed decision making
- Availability of data sources (volume, variety, velocity)
- Support traditional models with unorthodox evidence-based engineering

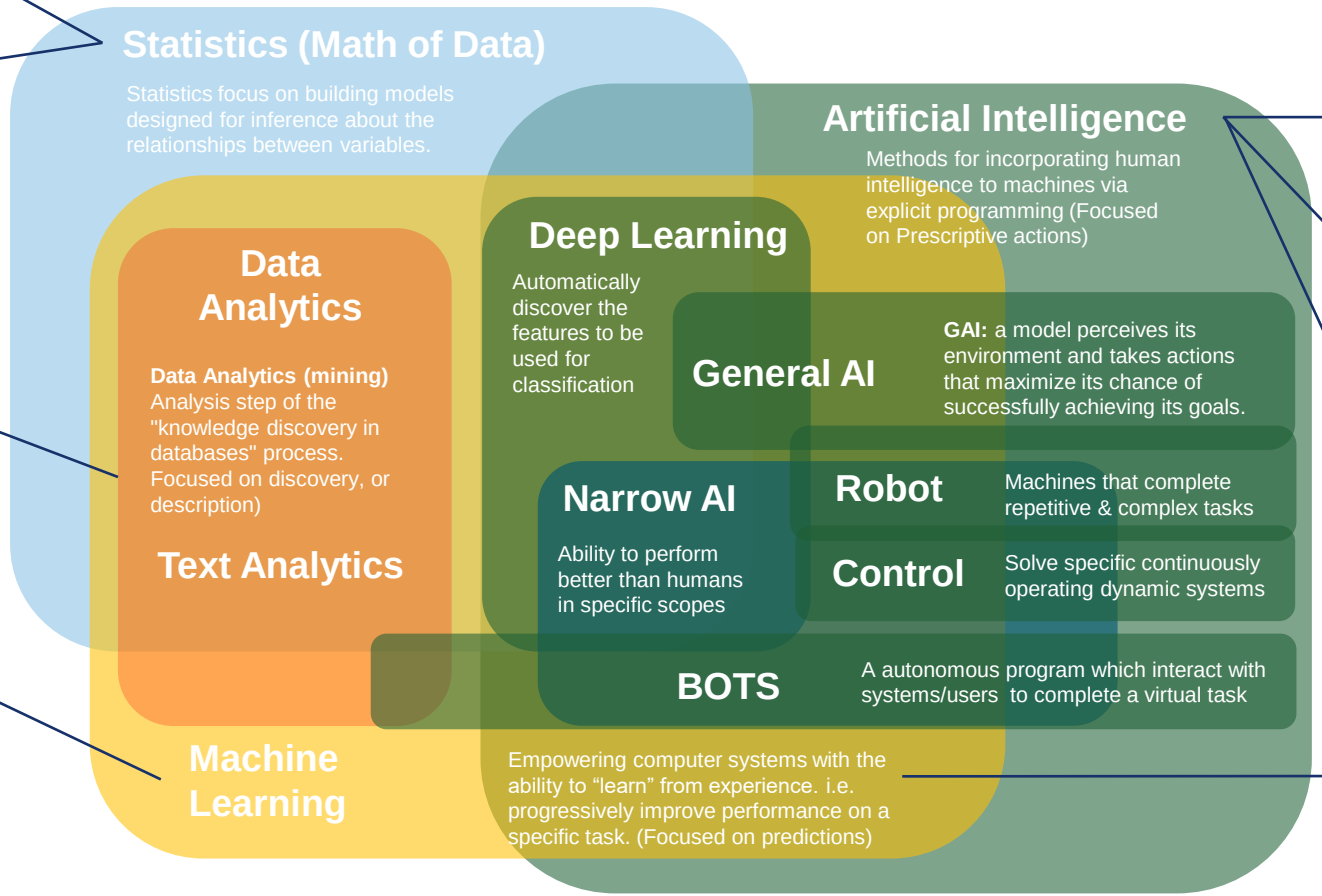
Data Science makes us aware of (data) relationships that were not seen before

Why Data Science Using Python?

- Scripting have always been the arm-extension of engineers
- If you have done some scripting in the past (such as VBA) → it will be very easy.
- Python's popularity has grown
  - Python has a relatively simple syntax, yet very powerful
  - Python easier to learn and have a large support community

Data Science raises fear among us because of knowledge about programming

From Statistics to AI



Statistical Learning (Anomaly detection, Association rule learning, Multivariate Regression)

Unsupervised and Supervised Classification & Cluster analysis (Learning classification, hierarchical, k-means, SOM, PCA, DBSCAN, etc.)

Text Mining and NLP (retrieval, categorization, clustering, concept/entity extraction, sentiment analysis, document summarization, taxonomies, entity relation modelling)

Supervised Learning (Artificial neural networks, Multivariate regression, Naïve Bayesian, Support Vector Machine)

Logic-based (Logic programming, automated reasoning)

Probabilistic reasoning (Bayesian Networks, Hidden Markov models, Kalman filter, Particle filter, Decision theory, and Utility theory)

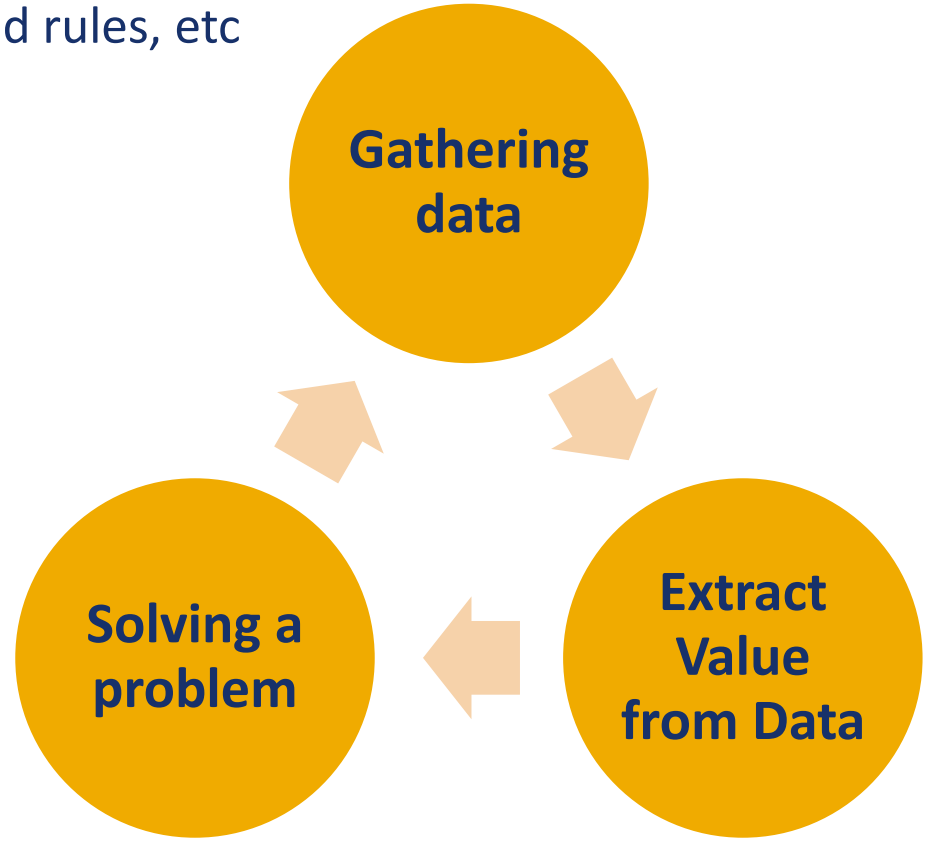
Knowledge-Based Systems (Case-based Reasoning, Expert Systems, Recommender systems (collaborative filtering)

Search and optimization (Simulated Annealing, Particle Swarm Optimization and Random Optimization)

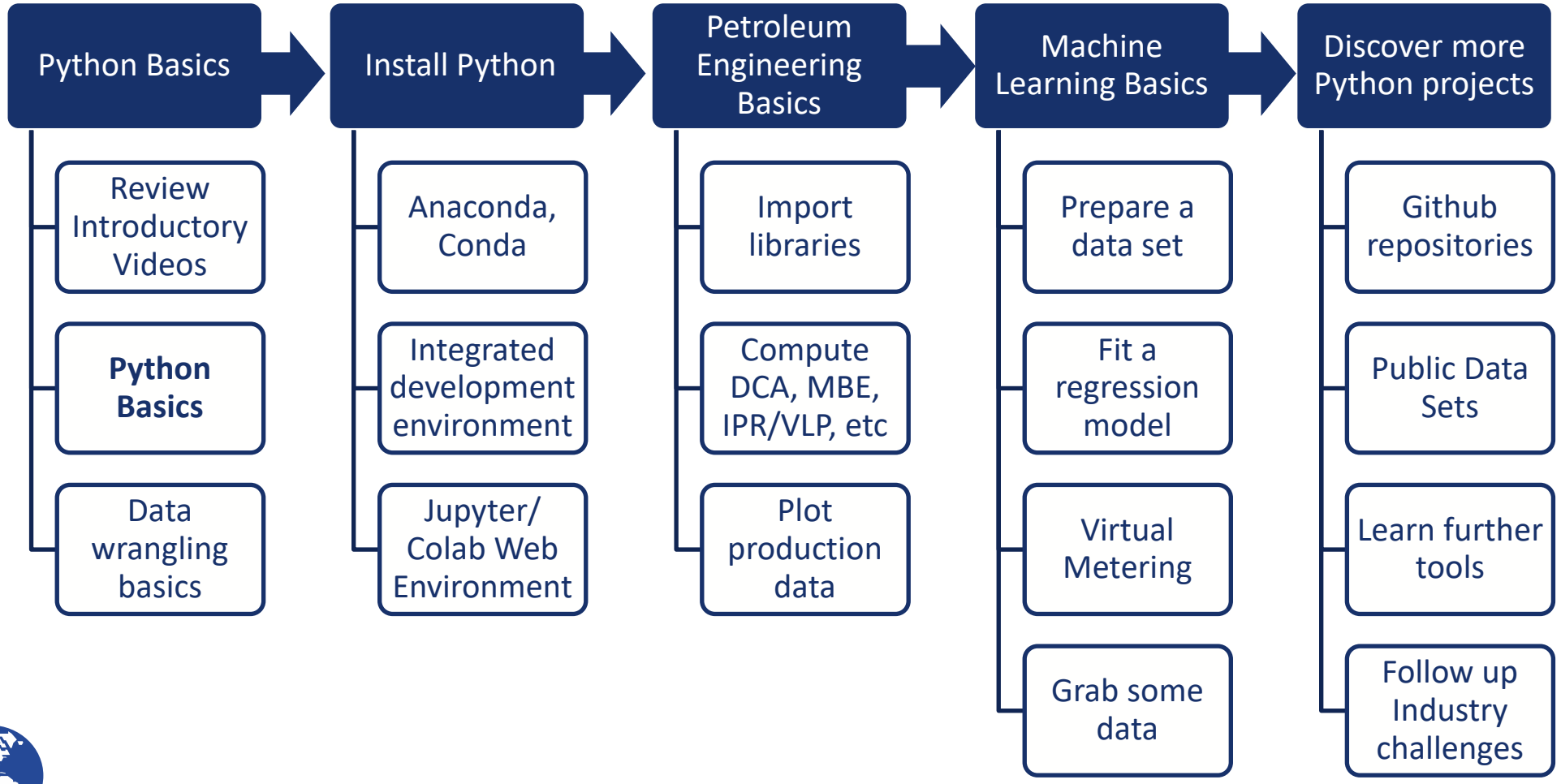
Data Science and Engineering Analytics

Solving a problem, discovering patterns and unexpected rules, etc

- Gathering data
  - numbers, text, images, sound
  - Private vs public.
- Extract Value from Data
  - Discovering patterns and unexpected rules
  - Reevaluating Hypothesis
- Solving a problem



A Roadmap to Kick Start in Python



Python Kick Start Program for PE

Python Basics

1. Print('hello'), Print(), /n
2. A=input('enter A=')
3. String to Numeric
4. String Count
5. Sentence Upper, Lower, Capitalize
6. Date and Time
7. Conditionals (if then elif)
8. Collections: Lists/arrays, dictionaries
9. Loops
10. Functions, Lambda
11. Managing the file system
12. Numpy
13. Create Pandas Data frame
14. Scikit Learn

Petroleum Engineering Basics

1. Decline Curve analysis
2. Material Balance Equation
3. Well Nodal Analysis (IPR/VLP)
4. Wellbore Pressure Response
5. Probabilistic Oil Reserves

Machine Learning Basics

1. Plot public data sets (Volve)
2. Define input and target (VM)
3. Split data for creating models
4. Clustering data
5. Model training and testing
6. Making predictions
7. Access public data sets

What is Python?

What is Python?	• Created by Guido van Rossum and first released in 1991 • Interpreted, flexible, general purpose programming Language • Human readable by design
Why Python?	• Easy to start • Ideal as an advance language • Extensive open community
What can I build with Python?	• Machine Learning Models • Artificial Intelligence Projects • Web Applications • Automation utilities

https://en.wikipedia.org/wiki/Python_(programming_language)

Installing Python

- Python is an Open Source, community supported programming language:
  - Interpreter Installation: https://www.python.org/downloads/
- IDE (integrated development environment)
  - **Spyder**: https://www.spyder-ide.org/
  - **PyCharm**: https://www.jetbrains.com/pycharm/
  - **Python for Visual Studio** (https://visualstudio.microsoft.com/vs/features/python/)
  - Many others: https://wiki.python.org/moin/IntegratedDevelopmentEnvironments
- **Anaconda** is an open source distribution of Python and R for data science → more than 1500 packages
- **Jupyter** and **Colab** are Open source, Web-based interactive development environment: software
  - Run inside JupyterLab on your laptop or in the cloud: **Jupyter Notebook**: https://jupyter.org/
  - Run in Colab environment in the cloud: **Colab** https://colab.research.google.com/

Python Basics: Print

- Syntax is straightforward, and easy to read (forcing indented blocks)
- Coding can start with only 16 statements (if, while/for, import…)

Print displays output to console

```
print('Hello world’)
```

```
Hello World
```

Getting information from the user

```
name = input('Please enter your name: ')
print(name)
```

```
Please enter your name: Tom
Tom
```

Printing blank lines and next line

```
# print() = prints a blank line
print()
# \n requests to prints a blank line in the middle of string
print(‘Hello World \nHello Everyone')
```

Debugging with print

```
print('Adding numbers')
x = 42 + 206
print('Performing division')
y = x / 0
print('Math complete’)
# This is a comment - it does nothing
```

Python Basics: String to Numeric

Numbers stored as strings must be converted to numeric values before doing math

```
first_num = input('Enter first number ')
second_num = input('Enter second number ')
print(int(first_num) + int(second_num))
print(float(first_num) + float(second_num))
```

```
Enter first number 50
Enter second number 60
110
110.0
```

Python Basics: Print Strings

Combine strings with +

```
your_name = input('What is your name? :')
your_age = input('Enter your age? : ')
print ('Hello ' + your_name.capitalize() +
        ', your age is ' + your_age)
```

```
What is your name? TOM
Enter your age? 25
Hello Tom, your age is 25
```

format strings that are saved to files and databases, or display to users

Format Text

```
sentence = 'Sample Sentence'
print(sentence.upper())
print(sentence.lower())
print(sentence.capitalize())
print(sentence.count('a'))
```

```
days_in_feb = 28
print(str(days_in_feb) + ' days in Feb')
```

Python Basics: Date and Time

Datetime objects to manipulate dates

```
# datetime and timedelta to define a period
from datetime import datetime, timedelta
today = datetime.now()
print('Today is ' + str(today))

# timedelta is used to define a period of time
one_day = timedelta(days=1)
yesterday = today - one_day
print('Yesterday was ' + str(yesterday))
```

```
Today is: 2021-05-15 12:53:51.831235
Yesterday was: 2021-05-14 12:53:51.831235
```

Converting it to a datetime allows you to use the date functions

```
from datetime import datetime, timedelta
birthday = input('Enter birthday (dd/mm/yyyy)? ')
b_date = datetime.strptime(birthday, '%d/%m/%Y')
print ('Birthday: ' + str(b_date))
birthday_eve = b_date - one_day
print('Day before birthday: ' + str(birthday_eve))
```

```
When is your birthday (dd/mm/yyyy)? 23/07/2000
Birthday: 2000-06-24 00:00:00
Day before birthday: 2000-06-23 00:00:00
```

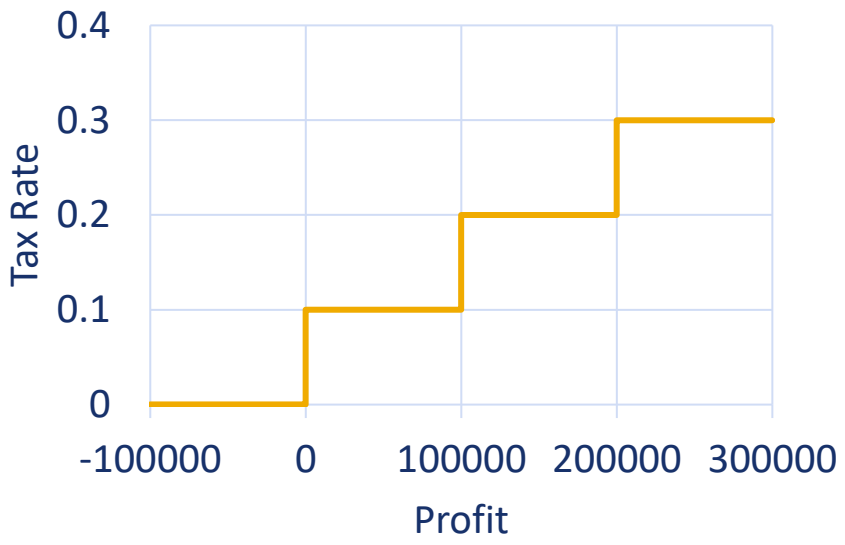
Python Basics: Conditionals

Different actions depending on conditions:

```
if Profit > 0.00:
    tax_rate = .15
else:
    tax_rate = 0
print(tax_rate)
```

Operation	Symbol
Less than	<
Less than or equal to	<=
Greater than	>
Greater or equal to	>=
Equal to	==
Not equal to	!=

```
if Profit >= 200000.00:
    tax_rate = .30
elif Profit >= 100000.00:
    tax_rate = .20
elif Profit > 0.00:
    tax_rate = .1
else:
    tax_rate = 0
print(tax_rate)
```




```
if Profit >= 100000.00 and Profit < 200000.00:
    tax_rate = .20
if Profit > 0.00 and Profit < 100000.00:
    tax_rate = .10
```

Python Basics: Collections: Lists

Lists: collections of items

```
countries = ['USA', 'UAE']
wells = []
wells.append(1202) # Add items to the end of list
wells.append(670)
print(countries)
print(wells)
print(wells[1]) # Collections are zero-indexed
```

```
['USA', 'UAE']
[1202, 670]
670
```

Lists are used to store anything, and any type:

```
names = ['Pepe', 'Juan']
names.append(2)
names.append(3.14)
print(names)
print(names[1])
print(names[3])
```

```
['Pepe', 'Juan', 2, 3.14]
Juan
3.14
```

Python Basics: Collections: Arrays

Arrays: collections of items

```
from array import array
wells = array('d')
wells.append(1202)
wells.append(670)
print(wells)
print(wells[1])
```

```
array('d', [1202.0, 670.0])
670.0
```

Arrays represent basic values and behave lists, except the type of objects stored in them is constrained. The type is specified by:

Type code	C Type	Minimum size in bytes
'b'	signed integer	1
'B'	unsigned integer	1
'u'	Unicode character	2
'h'	signed integer	2
'H'	unsigned integer	2
'i'	signed integer	2
'I'	unsigned integer	2
'l'	signed integer	4
'L'	unsigned integer	4
'q'	signed integer	8
'Q'	unsigned integer	8
'f'	floating point	4
'd'	floating point	8

Python Basics: Collections: List Ops

List Operations: count items – insert items

```
names = ['Tom', 'Carmen']
print(len(names)) # Get the number of items
names.insert(0, 'Mary') # Insert before index
print(names)
names.sort()
print(names)
```

```
2
['Mary', 'Tom', 'Carmen']
['Carmen', 'Mary', 'Tom']
```

List Operations: Retrieving ranges

```
names = ['Tom', 'Carmen', 'Mary’]
short_list = names[0:2] # Get the first two items
# Starting index and number of items to retrieve

print(names)
print(short_list)
```

```
['Tom', 'Carmen', 'Mary']
['Tom', 'Carmen']
```

Python Basics: Collections: Dictionary

Dictionary

```
person = {'first': 'Tom'}
person['last'] = 'Harrison'
print(person)
print(person['first'])
print(person['last'])
```

```
{'first': 'Tom', 'last': 'Harrison'}
Tom
Harrison
```

Dictionaries store key/value pairs; storage order is not guaranteed

```
well = {'Name': 'TX-001'}
well['API_No'] = '42-183-89876'
well['County'] = 'Fort Bend'
print(well)
print('Well name is ' + well['Name'])
print('Well API No. is ' + well['API_No'])
print('Well County is ' + well['County'])
```

```
{'Name': 'TX-001', 'API_No': '42-183-89876',
'County': 'Fort Bend'}
Well name is TX-001
Well API No. is 42-183-89876
Well County is Fort Bend
```

Python Basics: Loops

Loop through a collection

```
for name in ['Tom', 'Mary']:
    print(name)
```

```
Tom
Mary
```

Looping a number of times

```
for x in range(0, 5):
    print(x)
```

```
0
1
2
3
4
```

Looping with a condition

```
names = ['Mary', 'Tom', 'Carmen']
index = 0
while index < len(names):
    print(names[index])
    index = index + 1
```

```
Mary
Tom
Carmen
```

Python Basics: Functions

Simple function to get the initials

```
def get_initial(name):
    initial = name[0:1].upper()
    return initial
first = input('Enter first name: ')
first_initial = get_initial(first)
last = input('Enter last name: ')
last_initial = get_initial(last)
print('Initials are: ' + first_initial +
last_initial)
```

Parameter name is passed, and function is executed many times

```
Enter first name: Luigi
Enter last name: Saputelli
Initials are: LS
```

Python Basics: Functions

Simple function to print the duration of a process

```
from datetime import datetime
def print_time(task_name, time_start, time_end):
    difference = time_end - time_start
    total_seconds = difference.total_seconds()
    print(task_name + ' started on ' + str(time_start))
    print(task_name + ' ended on ' + str(time_end))
    print(task_name + ' completed in ' + str(total_seconds) + ' secs')
time_start = datetime.now()
for x in range(0,20000):
    print(x)
time_end = datetime.now()
print_time('Task 1', time_start, time_end)
```

3 parameters are passed, and function can be executed many times

```
Task 1 started on 2021-05-15 20:59:41.446098
Task 1 ended on 2021-05-15 20:59:43.515552
Task 1 completed in 2.069454 secs
```

Python Basics: Functions

Alternate way to estimated the duration of a process

```
# install requests
import requests
from timeit import default_timer

start_time=default_timer()
file_from_httpbin = requests.get('https://httpbin.org/delay/5').text
elapsed_time = default_timer() - start_time
print(f'Request access time is {elapsed_time:.2} secs')
```

```
Request access time is 5.83 secs
```

Python Basics: Lambda

Sort “items” in a Dictionary by “name”

```
members = [{'name': 'Tom', 'age': 41}, {'name': 'Juan', 'age': 62}]
members.sort(key=lambda item: item['name'])
```

```
[{'name': 'Juan', 'age': 62}, {'name': 'Tom', 'age': 41}]
```

Sort “items” in a Dictionary by length of “name”

```
members.sort(key=lambda item: len(item['name']))
```

```
[{'name': 'Tom', 'age': 41}, {'name': 'Juan', 'age': 62}]
```

Sort “items” in a Dictionary by “age”

```
presenters.sort(key=lambda item: item['age'])
```

```
[{'name': 'Tom', 'age': 41}, {'name': 'Juan', 'age': 62}]
```

Simple Lambda Function

```
# Simple y=x+3 function
(lambda x: x + 3)(3) # = 6
```

Python Basics: File System

Working with paths

```
# Import library
from pathlib import Path

# Get the current directory
cwd = Path.cwd()
print(cwd)

# Combine folder name and file name to create full path
new_file = Path.joinpath(cwd, 'file.txt')
print(new_file)
```

Some common libraries

- os
- pathlib
- math
- numpy
- pandas
- matplotlib
- scikit.learn
- keras
- tensorflow
- etc

Python Basics: Managing Files

Reading and Writing files

```
stream = open('demo.txt')
print(stream.readable()) # Can we read?
print(stream.read(1)) # Read the first character
print(stream.readline(5)) # Read any line
stream.close() # close the file

stream = open('output.txt', 'wt') # write text
stream.write('H') # write a single string
stream.writelines(['ello',' ','world']) # write multiple strings
stream.write('\n') # write a new line
names = ['Susan','Christopher'] # create a list of strings
stream.writelines(names) # write list of strings
stream.close() # close the stream (and flush data)
```

Python Basics: Pandas

Webinars

What is pandas?

- Open source/BSD-licensed library
- High performance data analysis tools
- https://pandas/pydata.org

Create a Data Frame

```
import pandas as pd
OilCo = pd.DataFrame([
    ['XOM','Irvin','USA'],
    ['Oxy','Houston','USA'],
    ['BP','London','United Kingdom'],
    ['Shell','The Haghe','Netherlands']],
     columns= ['Company','City','Country']
)
```

OilCo.**head(n)**- first *n* rows
OilCo.**tail(n)**- last *n* rows
OilCo.**shape** - dimensions
OilCo.**info** – detailed information

Familiar data structures and tools

Similar to a table in a database or spreadsheet

- Columns identified by name
- Rows of data
- Index column

Data Table

index	Name	City	Country
0	XOM	Irving	USA
1	Oxy	Houston	USA
2	BP	London	United Kingdom
3	Shell	The Hague	Netherlands

Read data from external csv files

```
OilCo = pd.read_csv('OilCompanies.csv')
```

Python Kick Start Program for PE

Webinars

Python Basics

1. Print('hello'), Print(), /n
2. A=input('enter A=')
3. String to Numeric
4. String Count
5. Sentence Upper, Lower, Capitalize
6. Date and Time
7. Conditionals (if then elif)
8. Collections: Lists/arrays, dictionaries
9. Loops
10. Functions, Lambda
11. Managing the file system
12. Numpy
13. Create Pandas Data frame
14. Scikit Learn

Petroleum Engineering Basics

1. Decline Curve analysis
2. Material Balance Equation
3. Well Nodal Analysis (IPR/VLP)
4. Wellbore Pressure Response
5. Probabilistic Oil Reserves

Machine Learning Basics

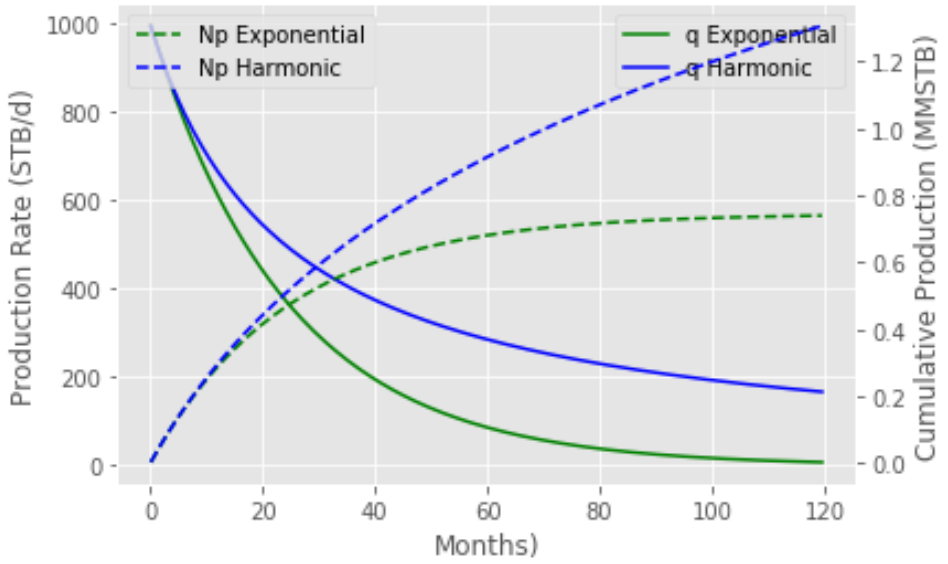
1. Plot public data sets (Volve)
2. Define input and target (VM)
3. Split data for creating models
4. Clustering data
5. Model training and testing
6. Making predictions
7. Access public data sets

SPE International

PE Basics: Decline Curve

```
import numpy as np
import matplotlib.pyplot as plt

# Input Data
t_1m = 1     # month
q_1m = 960   # stb/day
t_0m = 0     # month
q_0m = 1000 # stb/day
t = np.arange(0.1,120,0.5)

### Exponential Decline
b = np.log(q_0m/q_1m)/(t_1m - t_0m)
q = q_0m * np.exp( -b * t ) # Forecast
Np = (q_0m - q) * 365/(b *12)

### Harmonic Decline
b2 = (q_0m/q_1m - 1)/(t_1m - t_0m)
q2 = q_0m / ( 1 + b2 *t)
Np2 = (np.log(q_0m) - np.log(q2))
*((q_0m *365) /(12*b2))
```

```
# Plot Production and Cumulative vs Time (months)
ax = plt.subplot()
ax.plot(t, q, c = "g" ,  label= "q Exponential")
ax.plot(t, q2, c = "b" ,  label= "q Harmonic")
ax.set_xlabel("Months)")
ax.set_ylabel("Production Rate (STB/d)")
plt.legend()
ax2 = ax.twinx()
ax2.plot(t, Np/1000000, c = "g" , label = "Np Exponential", linestyle = "--")
ax2.plot(t, Np2/1000000, c = "b" , label = "Np Harmonic", linestyle = "--")
ax2.set_ylabel("Cumulative Production (MMSTB)")
plt.legend()
plt.grid(False)
```



https://github.com/Navdeep-Dhaka/Data-science-in-oil-and-gas

PE Basics: MBE Gas Cap (1/5)

$$F = N * (E_o + mE_g)$$

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Input Data
Pr = [3330,3150,3000,2850,2700,2550,2400] # psi
Np = [0,3.295,5.903,8.852,11.503,14.513,17.73] # MMSTB
Gp = [0,3460,6257,10268,14206,18359,23049] #MMSCF
Bo = [1.2511,1.2353,1.2222,1.2122,1.2022,1.1922,1.1822] # rb/stb
Bg = [0.00087,0.00092,0.00096,0.00101,0.00107,0.00113,0.0012] # rb/scf
Rs = [510,477,450,425,401,375,352] # scf/stb
Rsi = 510
Boi = 1.2511
Bgi = 0.00087
# Build Dataframe
df = pd.DataFrame({'Pr':Pr, 'Np':Np, 'Gp':Gp, 'Bo':Bo, 'Bg':Bg, 'Rs':Rs})

# Build Rp and add to df
df['Rp']= df.Gp/df.Np
# Build F, Eo and Eg and add to df
df['F'] = df['Np']*(df['Bo'] + (df['Rp'] - df['Rs']) * df['Bg']) # MMbbl
df['Eo'] = df['Bo'] - Boi + (Rsi - df['Rs'] ) * df['Bg'] # rb/stb
df['Eg'] = Boi *(df['Bg']/Bgi - 1) # rb/stb
```

https://github.com/Navdeep-Dhaka/Data-science-in-oil-and-gas

Build a Data Frame with Input Data

Index	Pr	Np	Gp	Bo	Bg	Rs
0	3330	0	0	1.2511	0.00087	510
1	3150	3.295	3460	1.2353	0.00092	477
2	3000	5.903	6257	1.2222	0.00096	450
3	2850	8.852	10268	1.2122	0.00101	425
4	2700	11.503	14206	1.2022	0.00107	401
5	2550	14.513	18359	1.1922	0.00113	375
6	2400	17.73	23049	1.1822	0.0012	352

Compute additional columns in the same Data Frame (Rp, F, Eo and Eg)

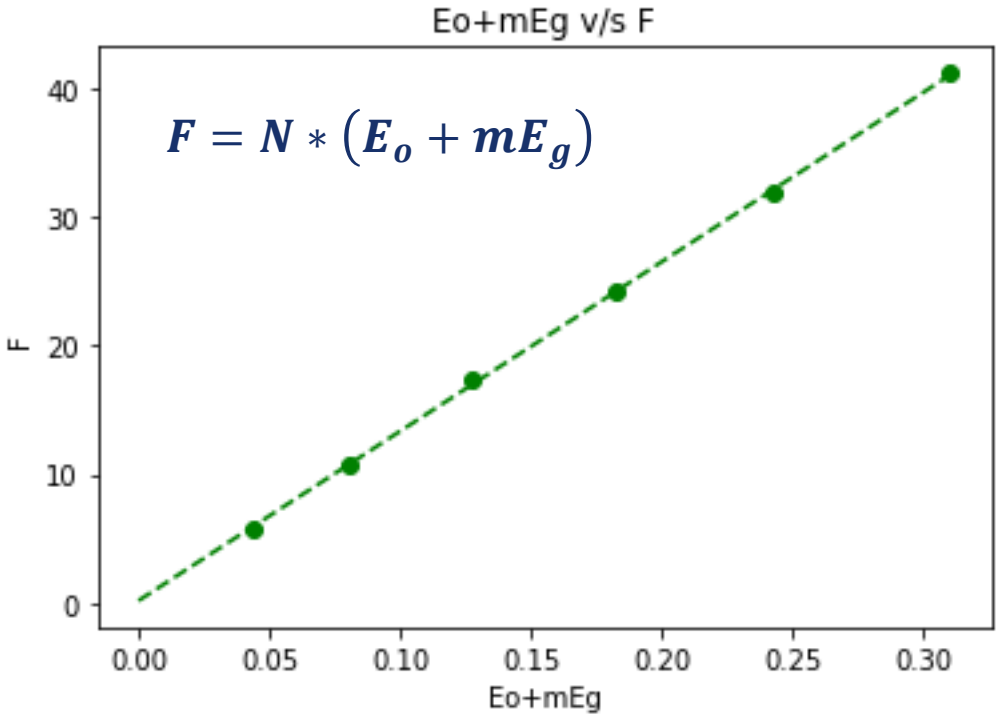
Rp	F	Eo	Eg
nan	nan	0	0
1050.08	5.80754	0.01456	0.0719023
1059.97	10.6713	0.0287	0.129424
1159.96	17.3014	0.04695	0.201326
1234.98	24.0937	0.06773	0.287609
1265	31.8982	0.09365	0.373892
1300	41.1301	0.1207	0.474555

PE Basics: MBE Gas Cap (2/5)

Case 1. N is unknown, m is known

```
# The slope of Eo+mEg v/s F curve give vale of OIIP
# Given m = 0.4
m = float(input('enter value of m = '))
mode = np.polyfit((Eo[1:7] + m*Eg[1:7]),F[1:7],1)

plt.plot(Eo + m * Eg, F, 'og')
plt.plot(Eo + m * Eg, mode[0]*(Eo + m * Eg) + mode[1] , 'g--')
plt.title('Eo+mEg v/s F' )
plt.xlabel('Eo+mEg')
plt.ylabel('F')
print(f'OIIP is {mode[0]:0.5} MMSTB for m = {m} ' )
```

```
enter value of m = 0.4
OIIP is 131.44 MMSTB for m = 0.4
```



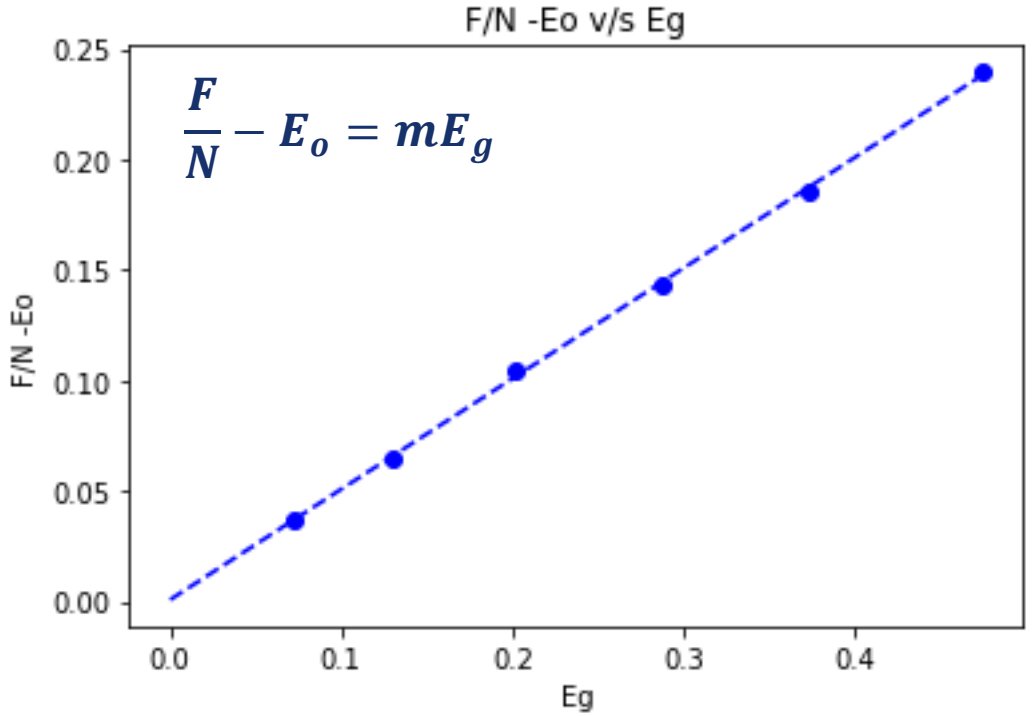
After https://github.com/Navdeep-Dhaka/Data-science-in-oil-and-gas

PE Basics: MBE Gas Cap (3/5)

Case 2. m is unknown and N is known

```
# The slope of (F/N - Eo) v/s Eg give value of m
# Given N is 114.2 MMSTB
N = float(input('enter value of OIIP in mmstb = '))
fit = np.polyfit(Eg[1:7],(F[1:7]/N - Eo[1:7]),1)

plt.plot(Eg, (F/N -Eo), 'ob')
plt.plot(Eg, fit[0]*Eg + fit[1], 'b--') # fit[1]~0
plt.title('F/N -Eo v/s Eg' )
plt.xlabel('Eg')
plt.ylabel('F/N -Eo')
print(f' m is {fit[0]:0.3} for {N} MMSTB OIIP' )
```

```
enter value of OIIP in mmstb = 114.2
 m is 0.5 for 114.2 MMSTB OIIP
```



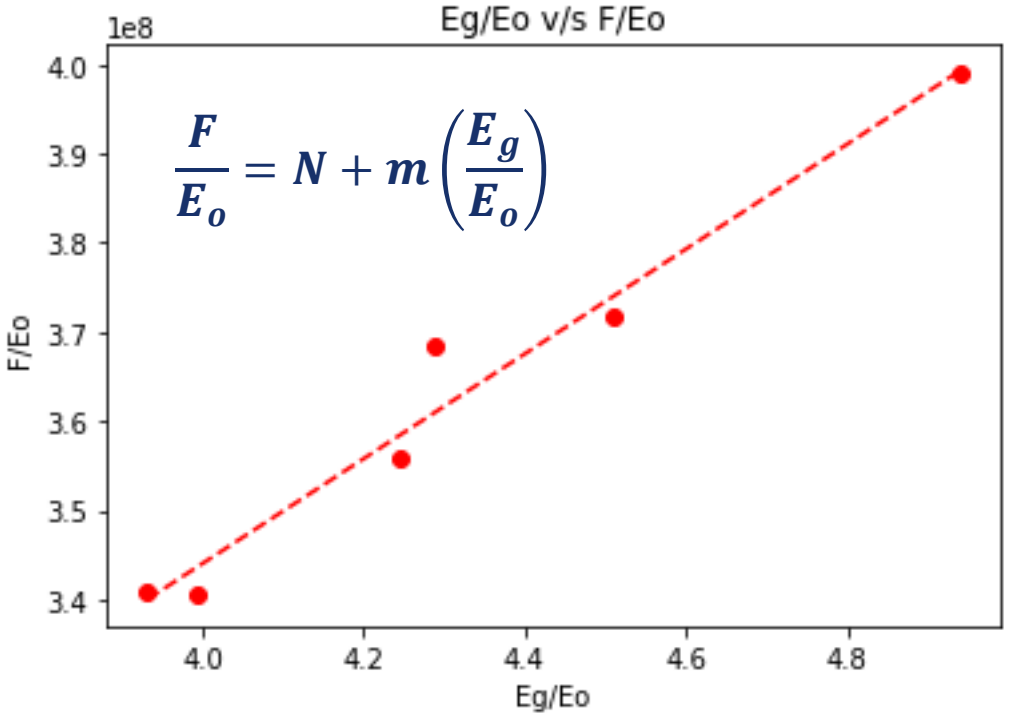
After https://github.com/Navdeep-Dhaka/Data-science-in-oil-and-gas

PE Basics: MBE Gas Cap (4/5)

Case 3. N and m are unknown

```
# The slope of (F/Eo) v/s Eg/Eo give value of m
df['F/Eo'] = df['F'] *10**6/df['Eo']
df['Eg/Eo'] = df['Eg']/df['Eo’]
model = np.polyfit(df['Eg/Eo'][1:7],df['F/Eo'][1:7],1)

plt.plot( df['Eg/Eo'], df['F/Eo'] , 'or')
plt.plot( df['Eg/Eo'], model[0]*df['Eg/Eo']+model[1] , 'r--')
plt.title('Eg/Eo v/s F/Eo')
plt.xlabel('Eg/Eo')
plt.ylabel('F/Eo’)

print(f'OIIP is {model[1]/1000000:0.5} MMSTB for
m={model[0]/model[1]:0.3}')
```

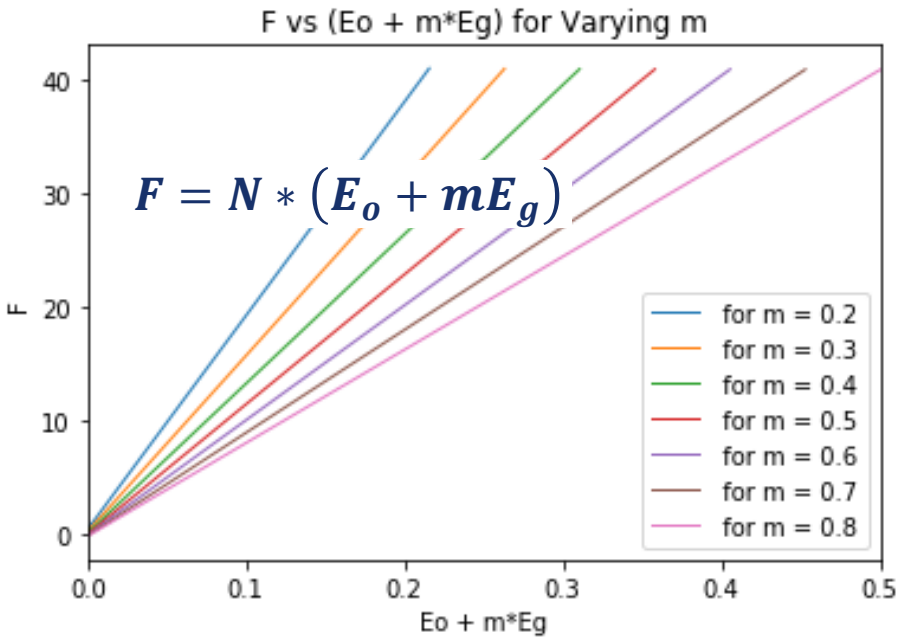
```
OIIP is 108.65 MMSTB for m = 0.542
```



After https://github.com/Navdeep-Dhaka/Data-science-in-oil-and-gas

PE Basics: MBE Gas Cap (5/5)

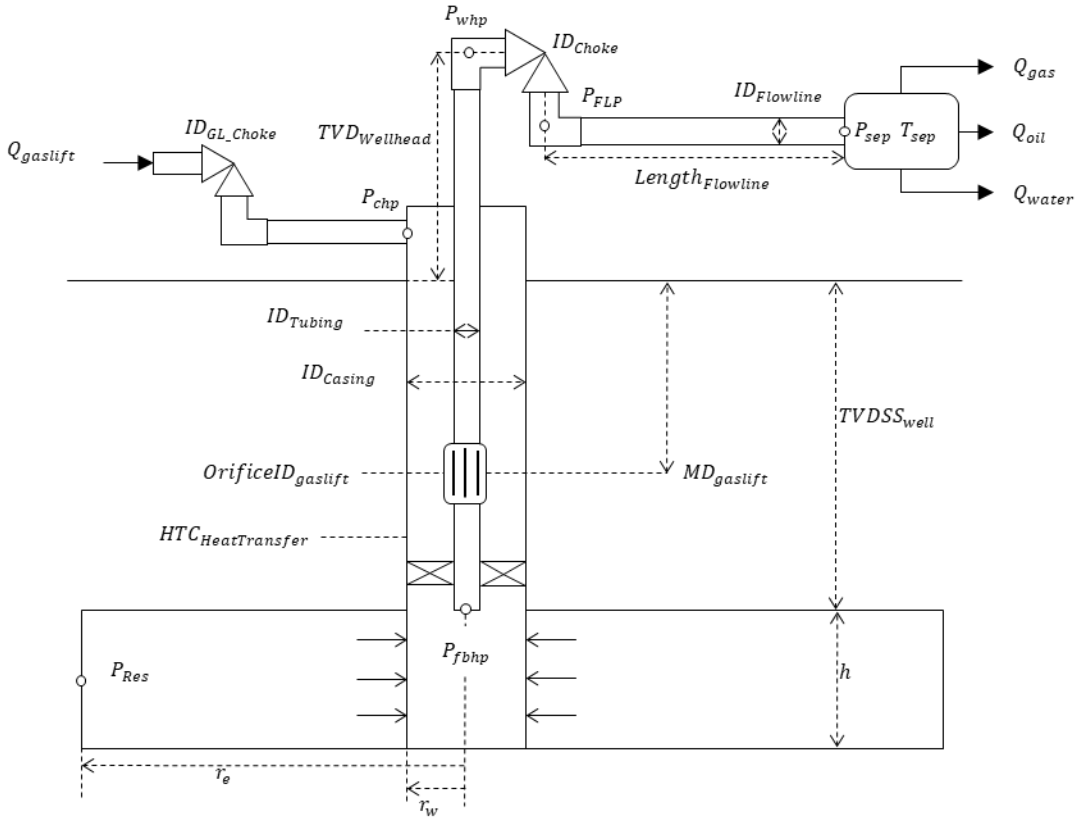
```
m = [0.2,0.3,0.4,0.5,0.6,0.7,0.8]
for i in m: df[f'Eo + {i}Eg'] = df['Eo'] + i * df['Eg']
plt.title('F vs (Eo + m*Eg) for Varying m')
plt.xlabel('Eo + m*Eg')
plt.ylabel('F')
plt.xlim(0,0.5)

for i in m:
    model = np.polyfit(df[f'Eo + {i}Eg'][1:7],df['F'][1:7],1)
    n_fitted = []
    for j in df[f'Eo + {i}Eg']:
        vv = model[0]*j + model[1]
        n_fitted.append(vv)
    plt.plot(df[f'Eo + {i}Eg'], n_fitted, label = f' for m = {i}')
    plt.legend()
    print(f'OIIP is {model[0]:0.4} MMSTB for m = {i} ' )
```



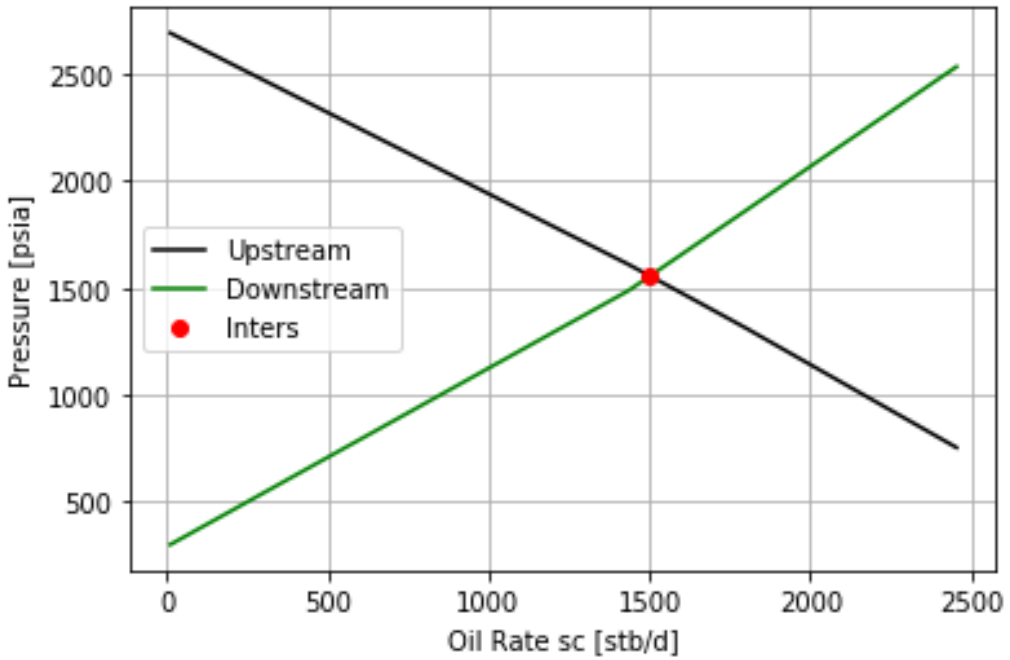
Index	Pr	Np	Gp	Bo	Bg	Rs	Rp	F	Eo	Eg	F/Eo	Eg/Eo	Eo + 0.2Eg	Eo + 0.3Eg	Eo + 0.4Eg	Eo + 0.5Eg	Eo + 0.6Eg	Eo + 0.7Eg	Eo + 0.8Eg
0	3330	0	0	1.2511	0.00087	510	nan	nan	0	0	nan	nan	0	0	0	0	0	0	0
1	3150	3.295	3460	1.2353	0.00092	477	1050.08	5.80754	0.01456	0.0719023	3.98869e+08	4.93834	0.0289405	0.0361307	0.0433209	0.0505111	0.0577014	0.0648916	0.0720818
2	3000	5.903	6257	1.2222	0.00096	450	1059.97	10.6713	0.0287	0.129424	3.71821e+08	4.50955	0.0545848	0.0675272	0.0804697	0.0934121	0.106354	0.119297	0.132239
3	2850	8.852	10268	1.2122	0.00101	425	1159.96	17.3014	0.04695	0.201326	3.68506e+08	4.2881	0.0872153	0.107348	0.127481	0.147613	0.167746	0.187879	0.208011
4	2700	11.503	14206	1.2022	0.00107	401	1234.98	24.0937	0.06773	0.287609	3.55732e+08	4.24641	0.125252	0.154013	0.182774	0.211535	0.240296	0.269056	0.297817
5	2550	14.513	18359	1.1922	0.00113	375	1265	31.8982	0.09365	0.373892	3.40611e+08	3.99244	0.168428	0.205818	0.243207	0.280596	0.317985	0.355374	0.392764
6	2400	17.73	23049	1.1822	0.0012	352	1300	41.1301	0.1207	0.474555	3.40763e+08	3.93169	0.215611	0.263067	0.310522	0.357978	0.405433	0.452889	0.500344

https://github.com/Navdeep-Dhaka/Data-science-in-oil-and-gas

PE Basics: Nodal Analysis



$$Q_{Liquid} = \left(\frac{k_{ro}}{\mu_o \beta_o} + \frac{k_{rw}}{\mu_w \beta_w}\right)\left(\frac{kh}{ln\frac{r_e}{r_w} + S}\right)(P_{Res} - P_{fbhp})$$



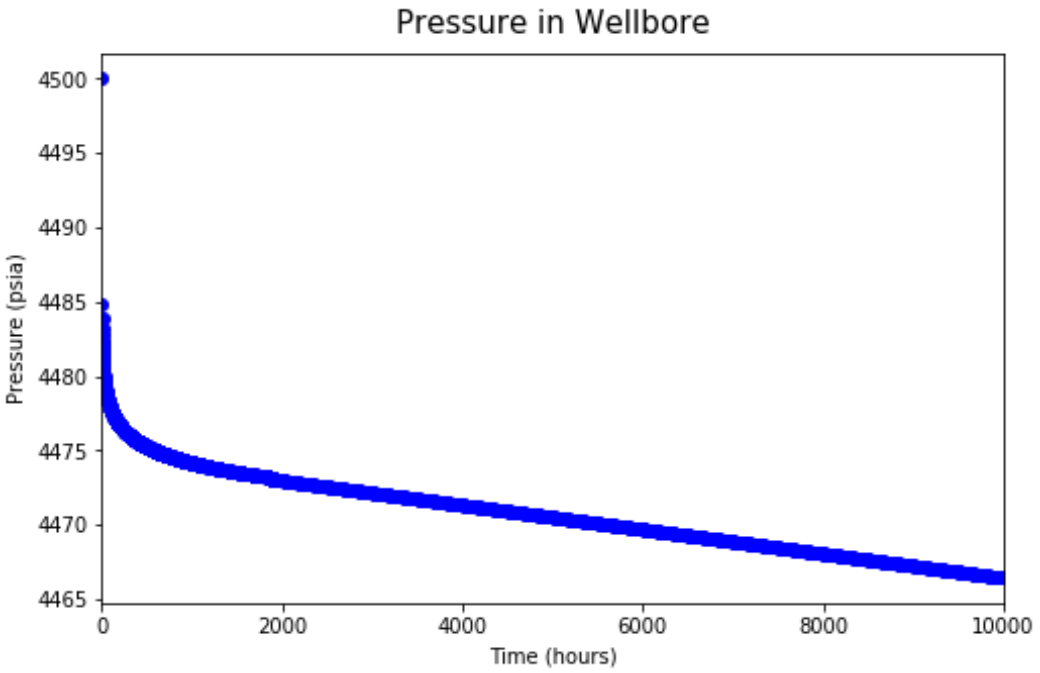
https://github.com/Apress/machine-learning-oil-gas-industry

PE Basics: Wellbore Pressure Response

Constant-Rate Solutions: Bounded Cylindrical Reservoir

```
r_eD = re / rw          # condition for finite acting (fa)
t_Dw = 0.25 * r_eD**2
t_fa = (poro * mu_oil * ct * (rw**2) * t_Dw)/(0.0002637 * k)
tsat = (t_Dw * poro * ct * (rw_conv**2))

if time[i] < t_fa:
   T_Dw = (0.000263*k*time[i])/(poro*mu_oil*ct*(rw**2))
        if T_Dw > 100:
           P_D = 0.5 *(((np.log(t_Dw)) + 0.80907))   # Eq 6.20
           Pwf = pi-((P_D*q*Bo*mu_oil)/(0.007082*k*h))
        if T_Dw < 100: P_D = "NaN“ ; Pwf = "NaN"
elif time[i] >= t_fa:
    T_Dw = (0.000263*k*time[i])/(poro*mu_oil*ct*(rw**2))
         if T_Dw > 25:
            P_D=(2*T_Dw/(r_eD**2))+np.log(r_eD)-0.75 # Eq 6.19
            Pwf = pi - ((P_D*q*Bo*mu_oil)/(0.007082*k*h))
         if T_Dw < 25: P_D = "NaN“ ; Pwf = "NaN"
```

The well becomes FINITE ACTING after 1896.09 hours



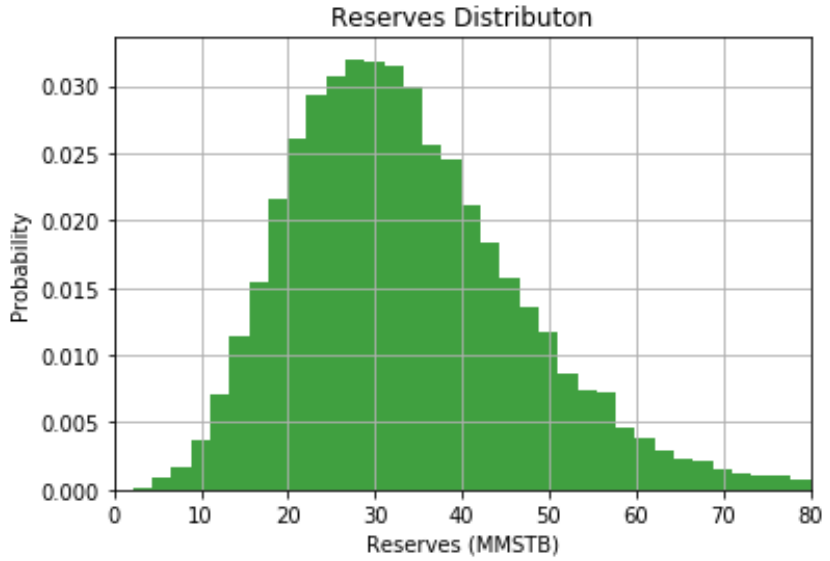

```
poro = 0.2     # Reservoir Porosity, fraction
ct = 20E-06    # Reservoir Compressibility, in psi^-1
k = 100        # Reservoir permeability, mD
rw = 8         # Wellbore radius, in
h = 1000       # Reservoir thickness, ft
mu_oil = 2     # Oil viscosity, cP
pi = 4500      # Reservoir initial pressure, psia
re = 5000      # Reservoir boundary radius, ft
q = 1000       # Oil flow rate, in STB/d
Bo = 1.1       # Oil FVF, in RB/STB
```

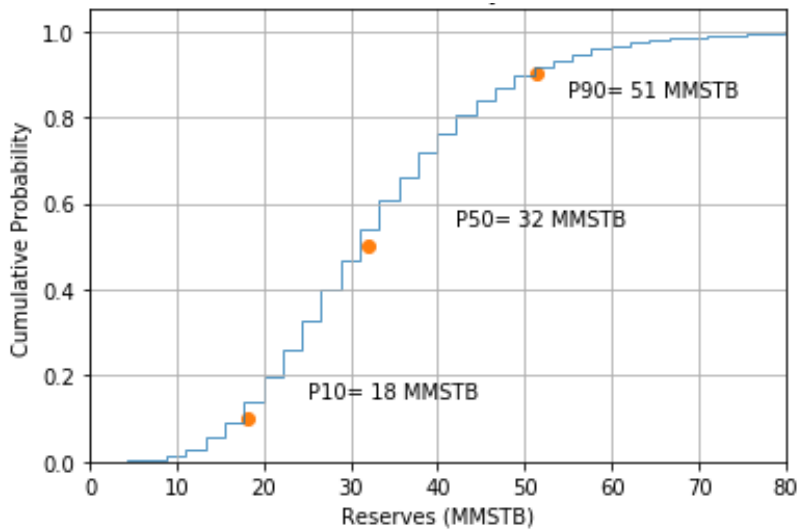
SPE Textbook Vol. 8 Fundamental Principles of Reservoir Engineering by Brian F. Towler
After Nuwara, 2020. https://github.com/yohanesnuwara

PE Basics: Probabilistic Reserves

Reserves calculations using the Monte Carlo simulation

```
from scipy.stats import norm, describe
import numpy as np

iterations = 10000
A = norm(640, 32).rvs(iterations)
h = norm(100, 20).rvs(iterations)
phi = norm(.2,.05).rvs(iterations)
Swi = norm(.25,.05).rvs(iterations)
NTG = norm(0.7,.2).rvs(iterations)
RF = norm(0.5,.1).rvs(iterations)
Bo = norm(1.1,.01).rvs(iterations)

Reserves = ( 7758 * A * h * phi * (1-Swi) * RF / Bo ) / 1e6
n, mm, m, v, sk, kurt = describe(Reserves, axis=0)

# Calculate the P90, P50 and P10 values
P10 = np.percentile(Reserves,10)
P50 = np.percentile(Reserves,50)
P90 = np.percentile(Reserves,90)
```





After Rémy Ghalayini. https://github.com/rghalayini

Python Kick Start Program for PE

Python Basics

1. Print('hello'), Print(), /n
2. A=input('enter A=')
3. String to Numeric
4. String Count
5. Sentence Upper, Lower, Capitalize
6. Date and Time
7. Conditionals (if then elif)
8. Collections: Lists/arrays, dictionaries
9. Loops
10. Functions, Lambda
11. Managing the file system
12. Numpy
13. Create Pandas Data frame
14. Scikit Learn

Petroleum Engineering Basics

1. Decline Curve analysis
2. Material Balance Equation
3. Well Nodal Analysis (IPR/VLP)
4. Wellbore Pressure Response
5. Probabilistic Oil Reserves

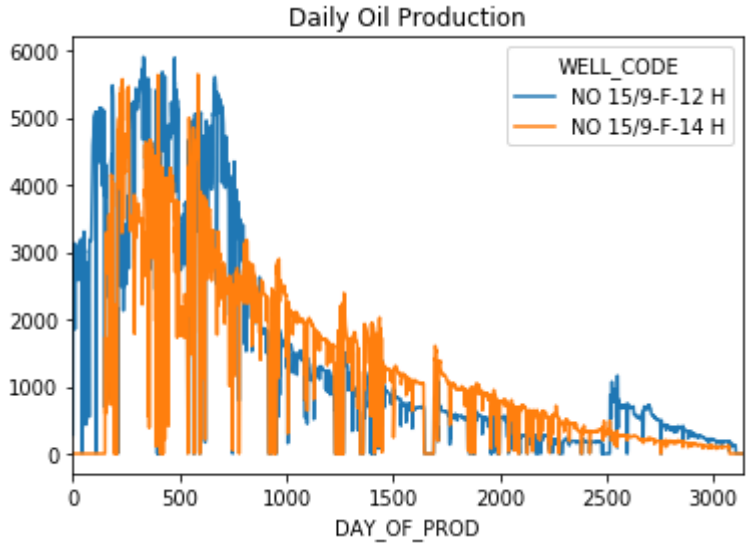
Machine Learning Basics

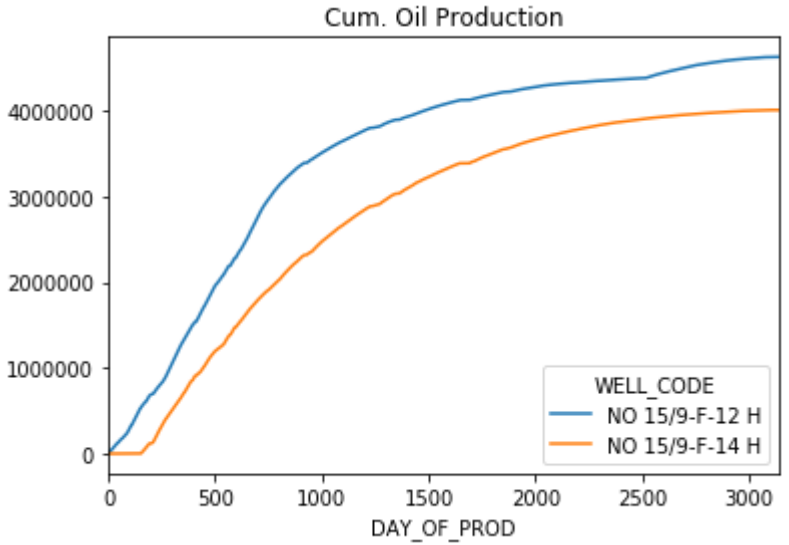
1. Plot public data sets (Volve)
2. Define input and target (VM)
3. Split data for creating models
4. Clustering data
5. Model training and testing
6. Making predictions
7. Access public data sets

Machine Learning Basics: Pandas Plot

Create a Data Frame

```
import pandas as pd
df = pd.read_csv('../Volve_Oil_Production.csv')
well_codes = df.WELL_CODE.unique()
df = df.pivot(index='DAY_OF_PROD', columns='WELL_CODE', values=['OIL_PROD_VOL', 'CUM_OIL_PROD'])
df.plot(y='OIL_PROD_VOL', title='Daily Oil Production')
df.plot(y='CUM_OIL_PROD', title='Cum. Oil Production')
```





https://www.equinor.com/en/what-we-do/digitalisation-in-our-dna/volve-field-data-village-download.html

Probability of future diminished production

Webinars

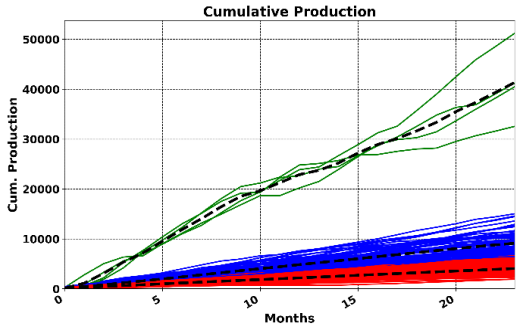
Daily Oil Production Data for many wells in a field or reservoir

Select Wells and time ranges for the analysis (e.g. 0 to 24 month and end of history status)

Well Status at the end of history, i.e. inactive wells, or Rate decline after 60 Months

Associate wells in each cluster and the well status at the end of history

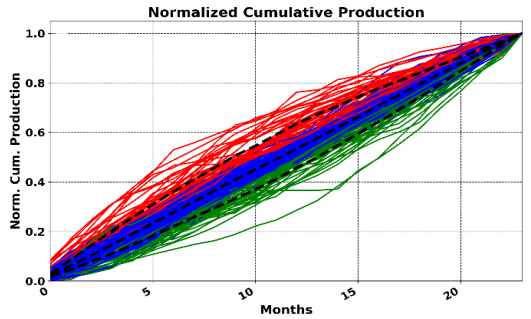
Probability of No Oil Production by Well after N Years later



Cumulative Oil Production Data

Cumulative Oil Production Clusters (e.g. Low, Medium, High Cumulative)

Combined Clusters generate 9 distinct cumulative-production categories → new data features



Normalized Cumulative Oil Production Data

Normalized Cumulative Oil Production Clusters (e.g. Decline, Steady, Increase)

Production Rate → / Cumulative Production ↓	Declining Rate	Steady Rate	Increasing Rate
Low	0.357	0.258	0.308
Medium	0.19	0.12	0.138
High	0.000	0.000	0.000
Average WCT & Dead Well Probability Correlation **Coefficient: 0.719561**			

SPE International

https://github.com/Apress/machine-learning-oil-gas-industry

Case Study: Data-driven Virtual Metering

$$\hat{Y} = \varphi(X) = Y_m + \varepsilon$$

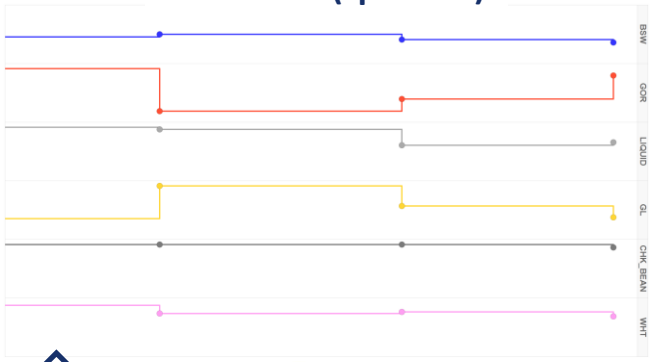
$$Liquid = f(BHP, WHP, WHT, GL)$$

$\hat{Y}$ is the virtual metering approximation function,
φ is the transfer function between measurements
X is the measurements input vector
Y_m is the labelled data (Well tests or measured pressure)
ε is the error

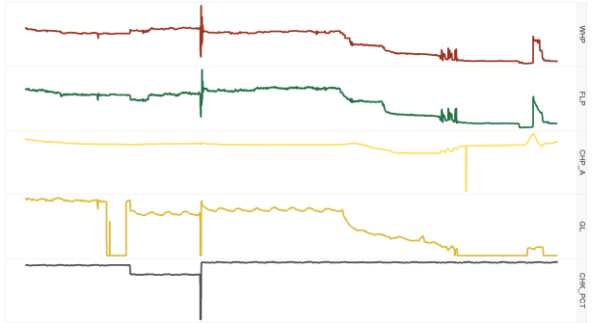
The objective of the data scientist is to find a suitable function φ including its parameters, that minimizes ε with reasonable amount of resources.

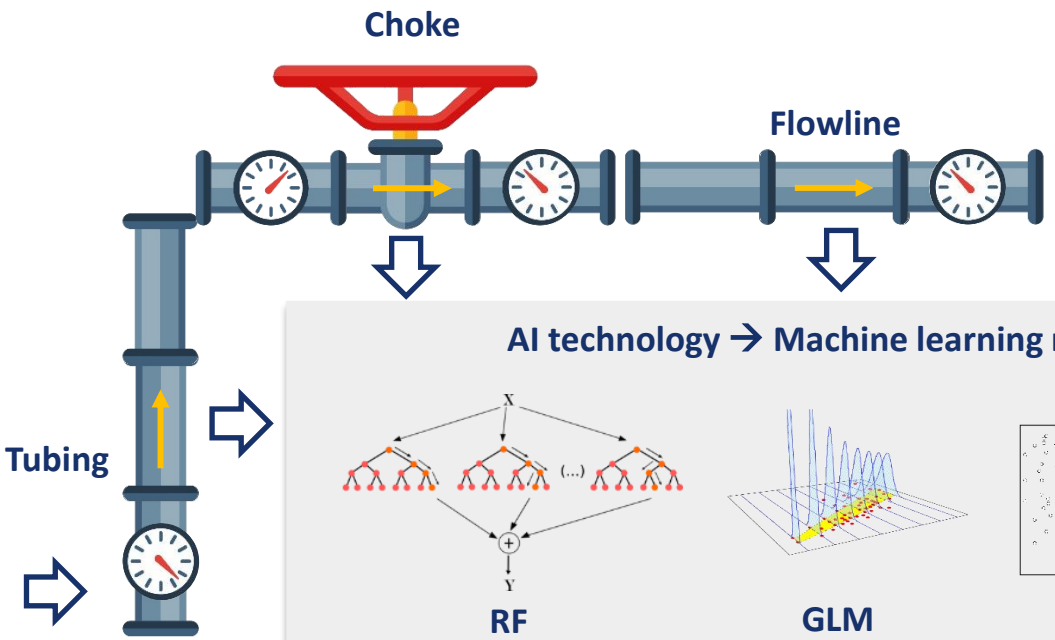
Gas-lift well Virtual Meter: How it works?

Flow Tests (sporadic)



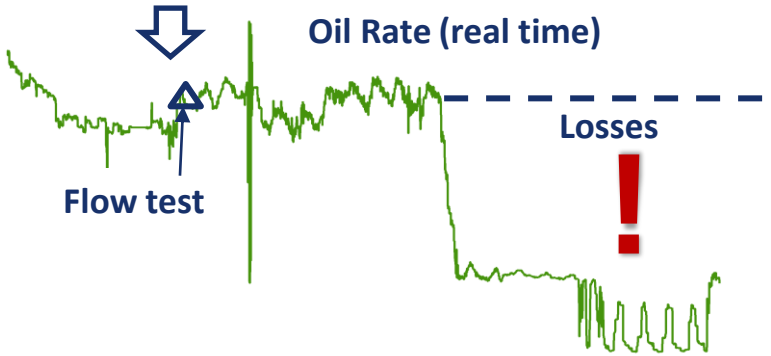
Pressures and Temperatures (real time)





AI technology → Machine learning models

Oil Rate (real time)



SPE-201696-MS

Machine Learning Basics: Virtual Meter (1/3)

Webinars

Given a set of Rate Measurements

Qoil	Qwater	Qgas	BHP	WHP	WHT	Tsep	Psep	Choke_in
954.6	0	2.39	5410.33	3185.75	83.3	60.32	100	0.25
801.9	200.5	2.01	5388	3015.38	86.88	60.65	100	0.25
634.7	423.2	1.59	5391.13	2808.8	90.83	61.19	100	0.25
448.1	672.2	1.12	5405.72	2515.99	95.11	61.97	100	0.25
238.9	955.7	0.6	5433.8	2059.65	99.88	63.1	100	0.25

Create a Data Frame

```
### Read data
import os
import pandas as pd
cwd = os.path.dirname(os.path.abspath(__file__))
df = pd.read_csv( cwd + "..\Well_Rates.csv" )
```

Define input and target

```
# Create input feature and target
X = df[["BHP", "WHP", "WHT", "Tsep", "Psep", "Choke_in"]]
y = np.array(df[["Qoil"]].values).reshape(-1, )
```

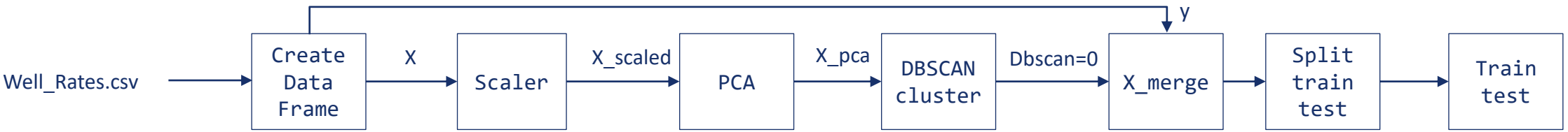
Split data for creating models

```
# Use train_test_split from scikit-learn
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.9, random_state=123)
```

SPE International

https://github.com/Apress/machine-learning-oil-gas-industry

Machine Learning Basics: Virtual Meter (2/3)




```
scaler = MinMaxScaler() ; X_scaled = scaler.fit_transform(X) # Scale Data
pca = PCA()
X_pca = pca.fit_transform(X_scaled) # Perform PCA
pca_df = pd.DataFrame(X_pca)

clustering = DBSCAN(eps=0.5, min_samples=12).fit(X_pca) # Cluster using DBSCAN
pca_df["DBSCAN"] = clustering.labels_ + 1
clust_df = pca_df[(pca_df["DBSCAN"] == 1)]  # Keep only the rows where DBSCAN=0
X_merge = X.merge(pca_df,left_index=True,right_index=True) # Split data
X_merge["Labelled Y"] = y

train_X, test_X, train_y, test_y = train_test_split(X.iloc[clust_df.index.tolist(),:],
y[clust_df.index.tolist()], test_size=0.5, random_state=123)
```

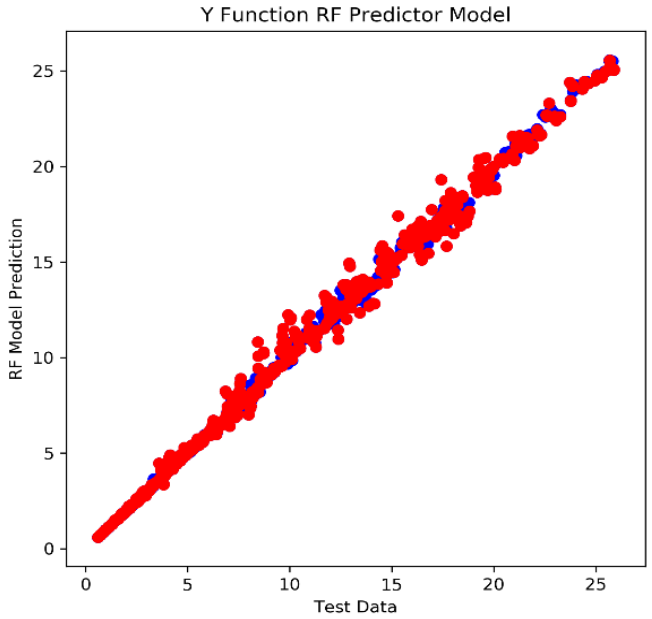
* DBSCAN: Density-Based Spatial Clustering of Applications with Noise

https://github.com/Apress/machine-learning-oil-gas-industry

Machine Learning Basics: Virtual Meter (3/3)

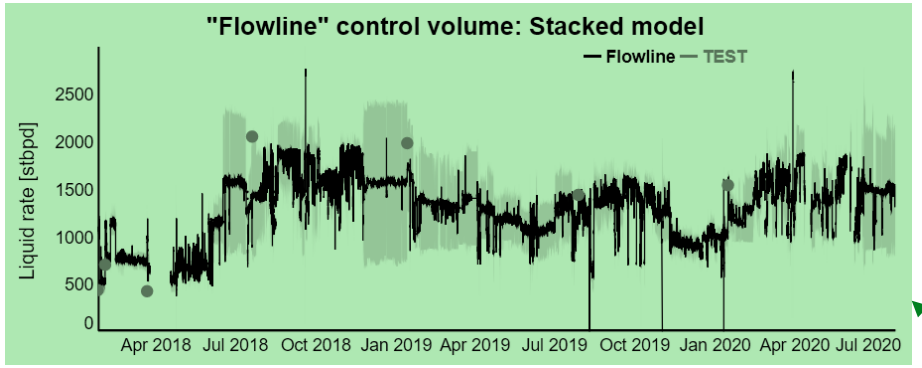
```
# Build Random Forestestimator:
model_RF = RF(n_estimators=200, random_state=123) # RF Regression
model_RF.fit(train_X, train_y)

# Predict for training and validation input sets:
pred_y_RF_test = model_RF.predict(test_X) # Blind test
pred_y_RF_train = model_RF.predict(train_X)
pred_y_RF_all = model_RF.predict(X.iloc[clust_df.index.tolist(),:])
r2_RF_train = r2_score(train_y, pred_y_RF_train) # Compute r2 scores
r2_RF_test = r2_score(test_y, pred_y_RF_test)
r2_RF_all = r2_score(y[clust_df.index.tolist()], pred_y_RF_all)
```

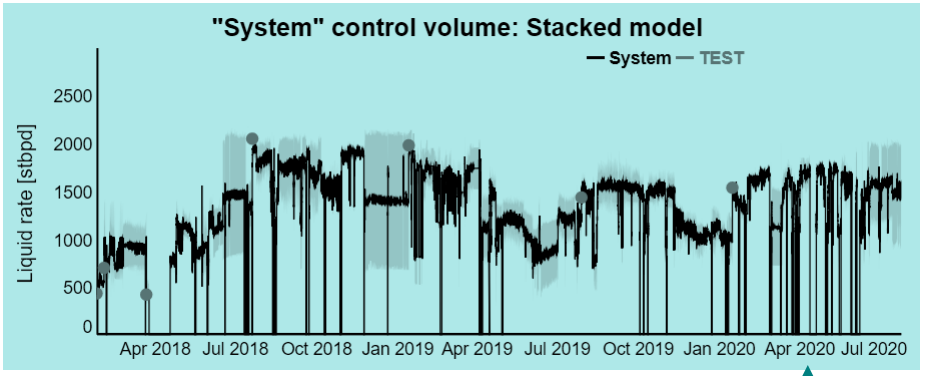
```
R^2 for train data Random Forest is 0.999169.
R^2 for blind test data Random Forest is 0.995007.
R^2 for all data Random Forest is 0.997096.
```

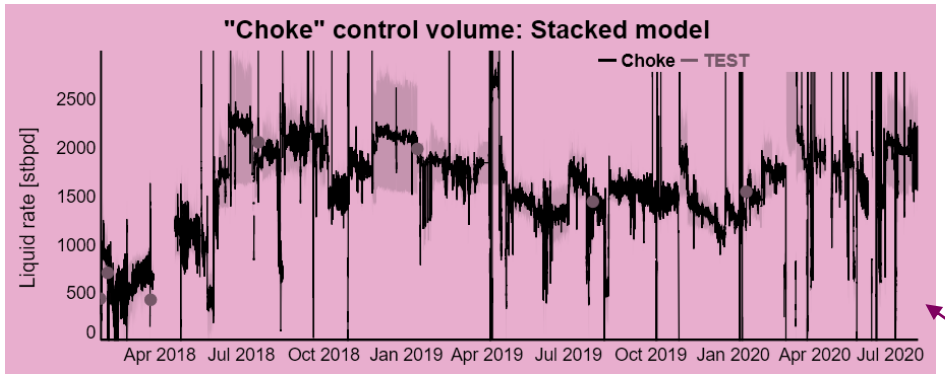


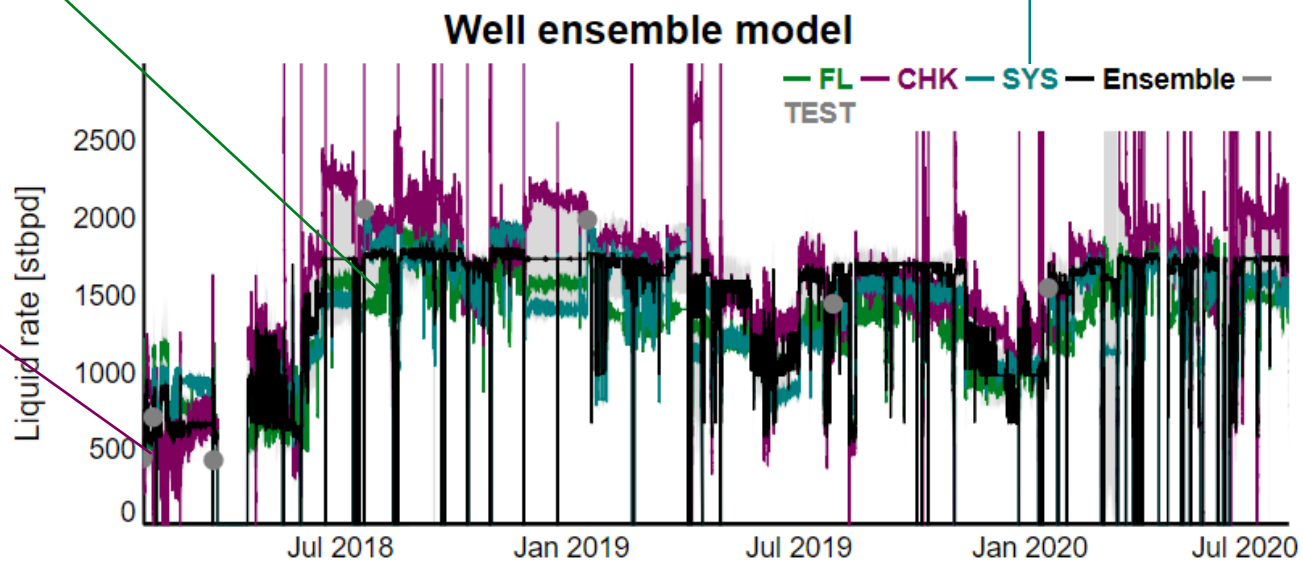
https://github.com/Apress/machine-learning-oil-gas-industry

Gas-Lift Well Virtual Meter - Ensemble model results









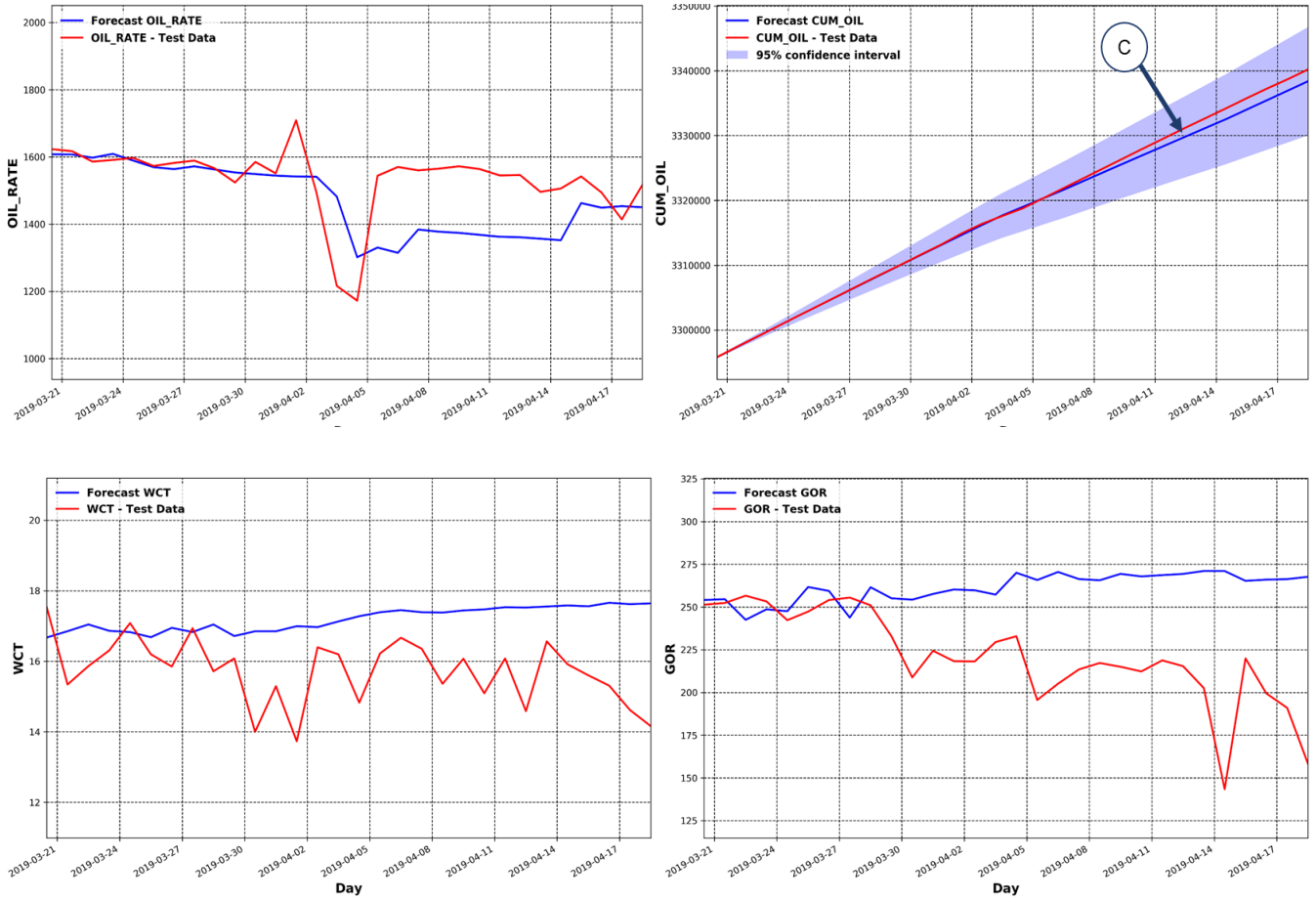
SPE-201696-MS

Gas-lift well Data Driven Model for Performance Forecasting

i. ARIMAX models order (14, 1, 0):
   i. 14 days auto-regression,
   ii. 1st order difference,
   iii. no moving average

ii. Choke, and gas-lift rates used as exogenous variables.

iii. Cumulative oil, cumulative water, and cumulative gas were modeled using ARIMAX.

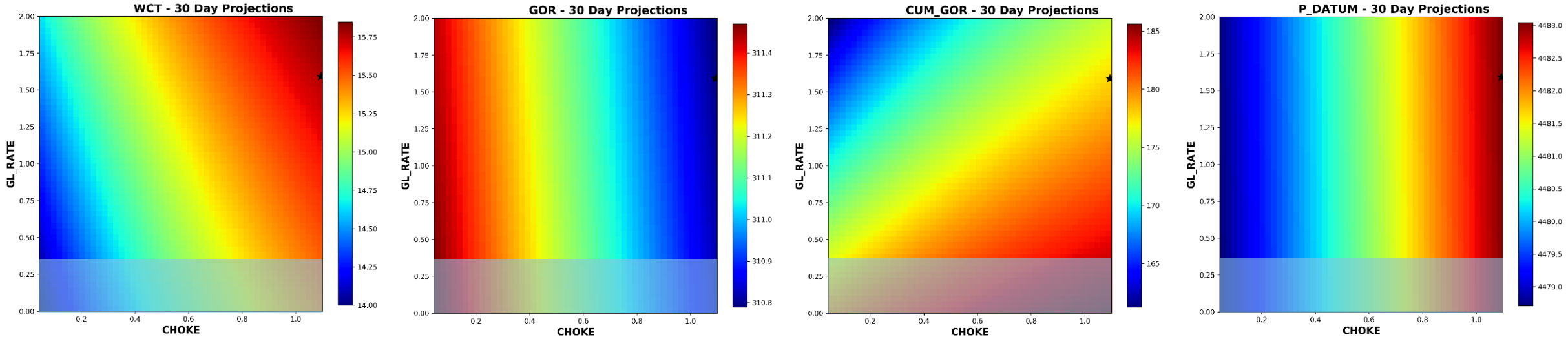


SPE-201696-MS

Gas-lift well Data Driven Choke vs Gas-lift rate sensitivity analysis

Webinars

Sensitivity analysis results showing 2D response surfaces for different variables including 30-day forecast of WCT, GOR, Cumulative GOR and Reservoir pressure.



"What-If" scenarios useful to answer questions such as:

- What would be the production if the well is operated at 100% choke open?
- What would be the production if gas injection was started for the well in question at a certain point of time?

SPE International

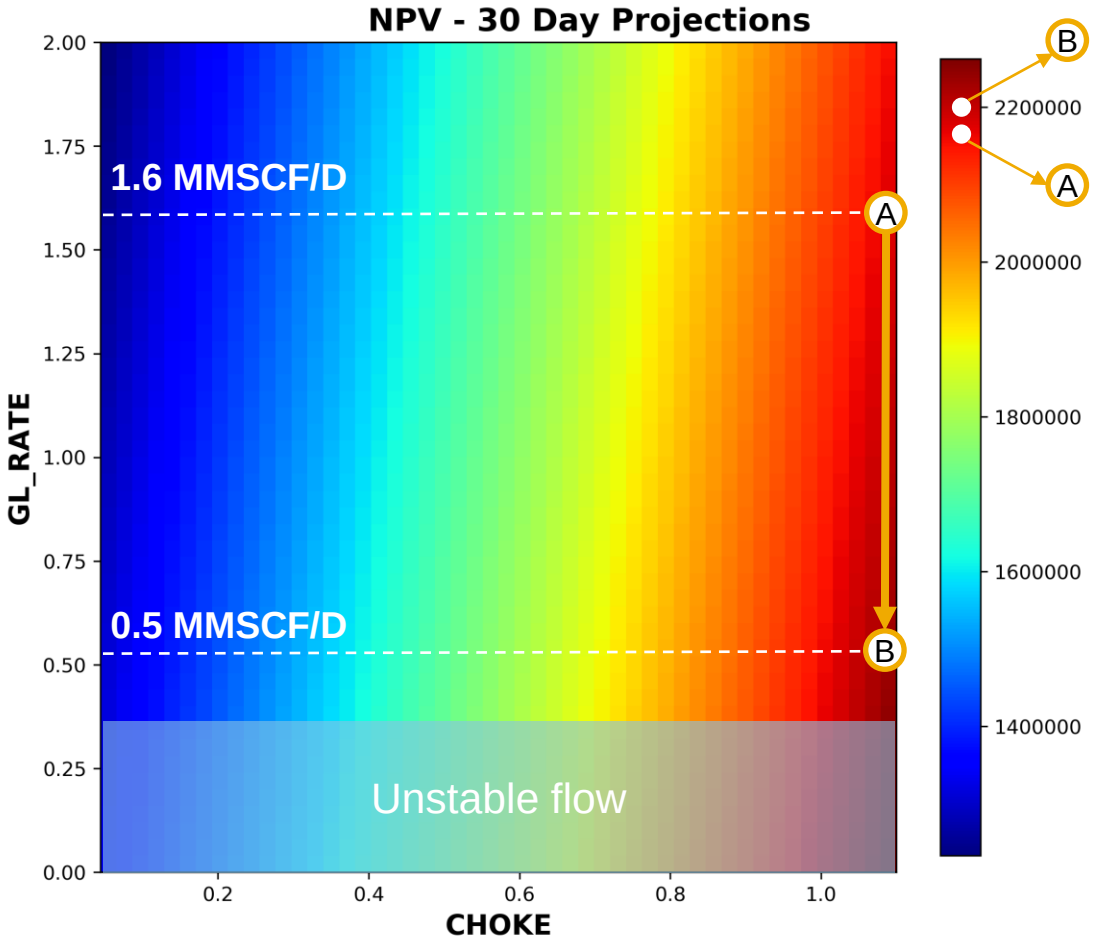
SPE-201696-MS

Gas Lift Well 30-day NPV Forecast and Optimization

- A simple NPV formula used

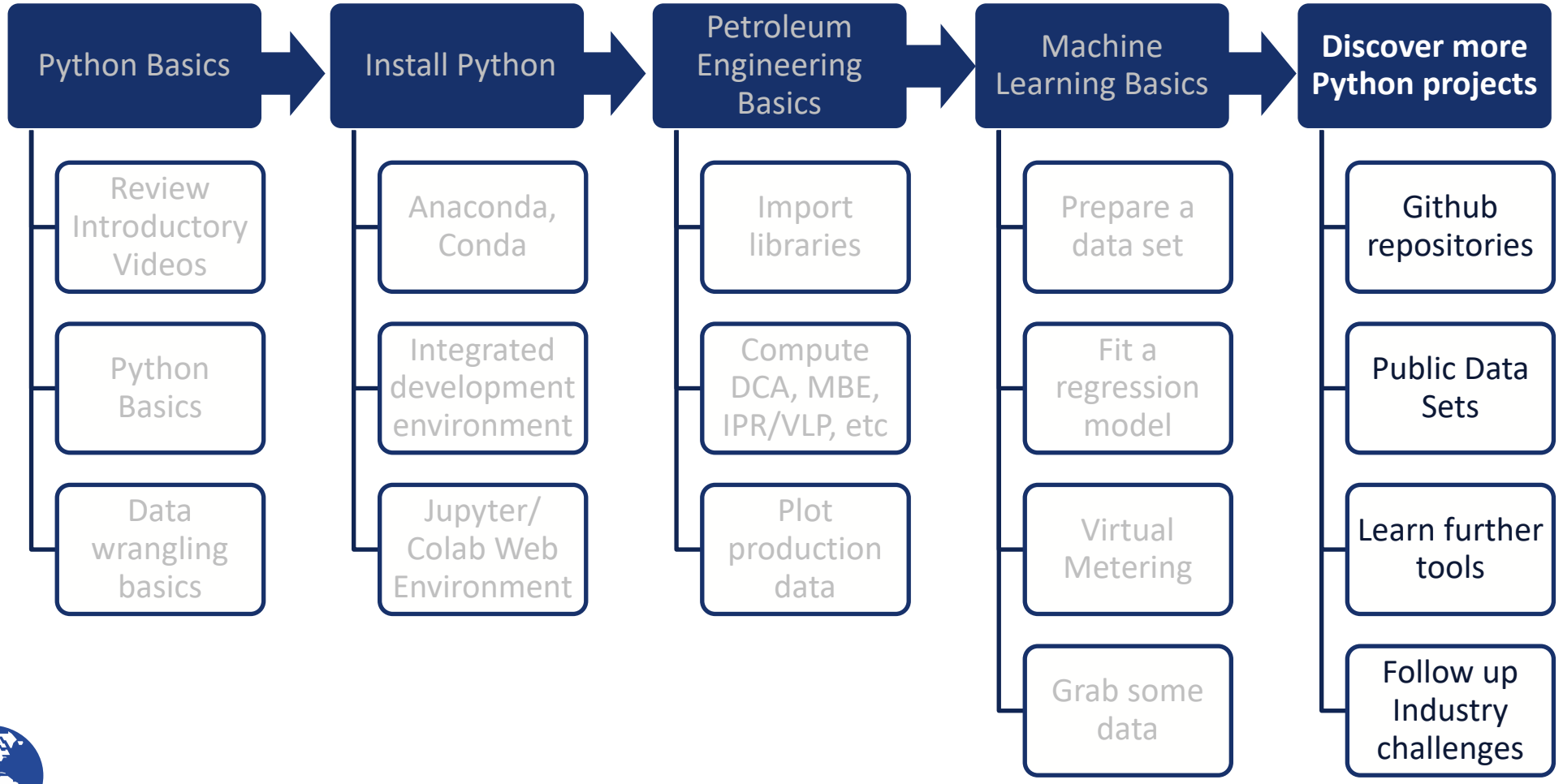
$$NPV = 50 * Np + 2500 * Gp - (500 * Gi_{gl} + 0.5 * Wp)$$

- Sensitivity analysis results showing 2D response surfaces for different variables.
- Shaded region is potential zone of unstable operations. Star shows current operating settings.
- Analysis suggests that the well is operating in near optimal zone.
- Slight decrease in the gas-injection rate increases NPV by ~0.05 MM$ in 30 days, or +2.5% in net profit gain.

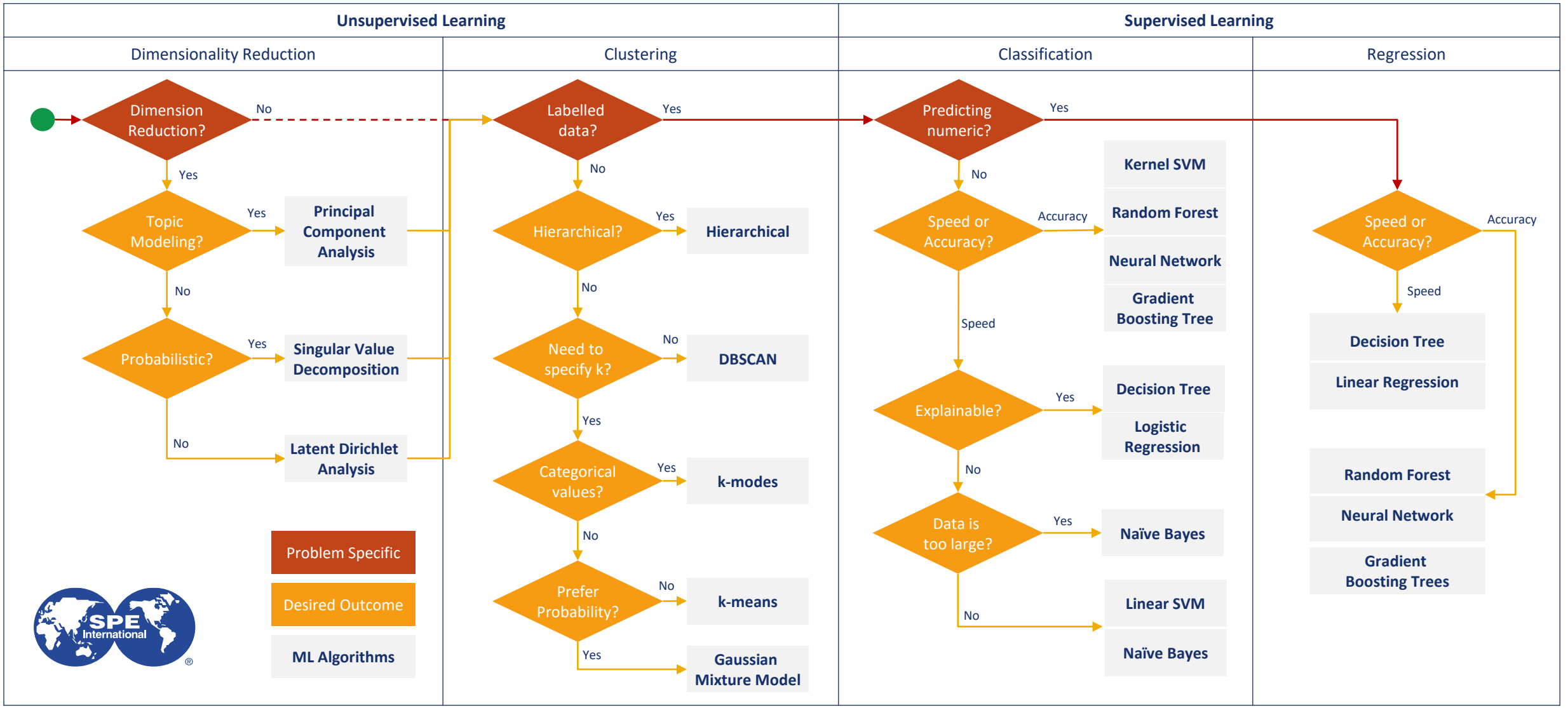


SPE-201696-MS

A Roadmap to Kick Start in Python



Data Analytics Approaches: how to start?



From Descriptive to Prescriptive

Applications	Well Opportunity Identification	ESP Analytics	Data Driven Reservoir Modeling	Drilling Analytics	Petrographic Image Analysis
Descriptive – Focus on gathering facts about past performance. Analytical dashboards with read-only data and high-level metrics.	How does well performance rank against targets?	How does ESP perform against targets?	What is reservoir past performance model? How does reservoir perform against targets?	How does well drilling perform against targets?	What are dominant lithofacies and reservoir performances?
Diagnostic - Make use of data to understand the reasons behind the observed values.	What are the precursor events to well performance?	What are the precursor events to ESP failure?	How does collected data aligns with reservoir model? Why does reservoir performance is as-is?	What are the causes of NPT?	How does lithofacies and completion affect well success?
Predictive – Forecasting to investigate decision influencing the process with analytical models.	How will the well perform if intervention is performed today?	When will the ESP fail?	How will the reservoir perform if new well is added? Or if VRR targets are change?	How will drilling performance evolve at current conditions?	How will the well perform for various location direction and completion?
Prescriptive – Recommend decisions that improve performance. Decision support systems cooperate with human.	Well will improve 30% performance if WSO is performed.	ESP will improve performance if frequency is changed	Reservoir will improve performance if certain plan is implemented: Rate changes, infill drilling, etc	Drilling will improve if WOB/ROP changes are implemented	The optimum well completion and direction are…

100’s of available use cases

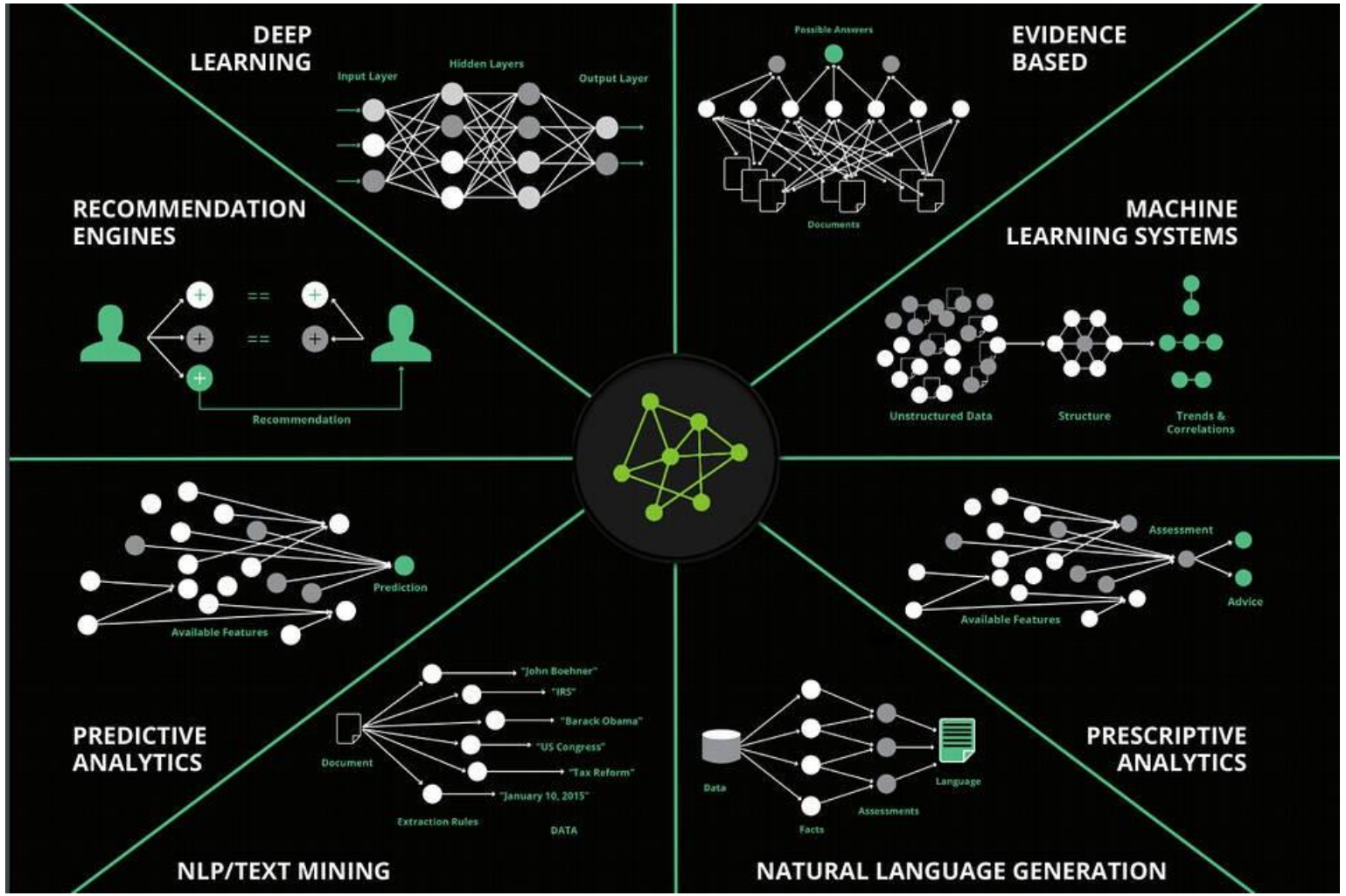
Webinars

General Tools	Geostatistics & Geology	Petroleum Engineering	Petroleum Engineering	Other Calculations
• Montecarlo Simulation	• Bayes Theorem to Geostatistical Mapping	• **Flow spinner calibration**	• Decline Curve Analysis	• Parameter Finding in 2D prblems
• Financial cash flow on DCA, monthly preview, Excel Export	• Multidimensional Scaling	• **Effect of Skin on Pressure Drawdown**	• Gas Cap Material Balance Equation	• Network modeling using Watts-Strogatz model
• Automatic Summarization of Technical Literature on Oil and Gas	• Collocated Cokriging	• **Effect of Viscosities on Water Cut**	• Deep Learning to Petroleum Well Production Forecasting Data	• 4 Phase Flowing Bottom Hole Pressure Calculation
• Get datasets from public sources	• Geostatistical Tools: variogram, confidence, interval prediction, etc	• **Vogel's and Fetkovich's Model for IPR;**	• Volve Field - Well production data Forecast	• Random walk with Obstacles
• Prediction of US Petroleum Refineries location	• Image analysis with PCA and Naive Bayes	• **Klink Enberg Effect**	• Estimating Future IPR from Reservoir Pressure Change	• Gas Lift Transient Models
• Openserver Connectors	• Unsupervised Classifier for lithology determination	• **Well Test**	• Fit Arps decline curves for oil and gas	• Leakage Analysis and Equipment Failure Detection
	• Statistically Valid and Significant Correlation	• **Oil and Gas Material Balance**	• Brent Prices Forecast	• Oil and gas production - Water analysis - PA
	• Reserves Probabilistic Estimation using Monte Carlo Simulation	• **Degradation modeling of piggable oil and gas pipelines corrosion**	• Flow in Porous Media PDE solving with Numeric Techniques	• Online Public Data Exploratory Analysis
		• **Oil and Gas Occurrence based on source rock parameters**		

https://github.com

AI Toolkit

- Deep learning
- Recommendation Engines
- Predictive Analytics
- NLP / Text Mining
- Evidence bases Learning
- Machine learning systems
- Prescriptive Analytics
- Natural Language generation



https://blog.else-corp.com/2017/10/summary-on-ai-algorithms/

Public Oil, Gas & Energy Data Sets

Volve Field Data	https://www.equinor.com/en/what-we-do/digitalisation-in-our-dna/volve-field-data-village-download.html
FracFocus (chemicals, additives, vols)	http://fracfocus.org/
West Texas Lands (land and mineral)	http://www.utlands.utsystem.edu/
Alaska O&G Conservation Commission	https://www.commerce.alaska.gov/web/aogcc/Data.aspx
California Div. of O&G and Geothermal	http://www.conservation.ca.gov/dog/Pages/Index.aspx
Colorado O&G Conservation Commission	http://cogcc.state.co.us/
Oklahoma Corporation Commission	http://www.occeweb.com/og/oghome.htm
Texas Railroad Commission	http://www.rrc.state.tx.us/
Wyoming O&G Conservation Commission	http://wogcc.state.wy.us/
USGS National Map Viewer	http://viewer.nationalmap.gov/viewer/
Energy Information Agency	https://www.eia.gov/totalenergy/data/browser/
Department of Energy	https://www.energy.gov/data/open-energy-data
Data World (misc)	https://data.world/datasets/energy

Awesome Reservoir Engineering

https://github.com/mralbu/awesome-reservoir-engineering

Webinars

Poseidon NW Australia SD seismic & well logs	https://drive.google.com/drive/folders/0B7brcf-eGK8Cbk9ueHA0QUU4Zjg
World Stress Map – Crustal stress field	http://www.world-stress-map.org/
NOPIMS – Open petroleum geoscience data	https://nopims.dmp.wa.gov.au/nopims/
UK National Data Repository	https://ndr.ogauthority.co.uk/dp/controller/
Athabasca Oil Sands Well Dataset	https://ags.aer.ca/publication/spe-006
ICGEM – Hosts gravity field spherical	http://icgem.gfz-potsdam.de/home
TerraNubis –*Open Seismic Repository*	https://terranubis.com/datalist/free/
Quantarctica: QGIS basemap for Antarctica	https://www.npolar.no/quantarctica/
Digital Rocks Portal porous micro-structures	https://www.digitalrocksportal.org/
GSQ Open Data Portal –Queensland resource	https://geoscience.data.qld.gov.au/
GSQ GitHub Repository	https://github.com/geological-survey-of-queensland
Geoscience Australia Portal	https://portal.ga.gov.au/
SARIG – South Australian Resources	https://map.sarig.sa.gov.au/
SEG Open Data Catalog	https://wiki.seg.org/wiki/Open_data

Selected Industry Challenges

Department of Energy	• http://energychallenge.energy.gov/ • https://www.energy.gov/data/articles/think-outside-box-during-our-open-data-design-contest
Kaggle	• https://www.kaggle.com/tags/energy
Production Forecasting	• https://www.kaggle.com/c/datascienceatraisa
Predicting Oil Share Price	• https://www.kaggle.com/javierbravo/a-tour-of-the-oil-industry?scriptVersionId=1445335&cellId=1
Analytics Vidhya	• https://datahack.analyticsvidhya.com/contest/all/
SPE Colombia Hackathon Oil & Gas	• https://www.spe.org.co/hackathon-oilgas/

Some great help resources

Webinars

ML Implementations in O&G - Success Stories & Challenges Ahead	https://seg.org/Events/SEG-Live/session/machine-learning-implementations-in-o-and-g
Cegal	https://www.youtube.com/user/bluebackreservoir https://github.com/cegaldev
Orkahub Energy	https://www.youtube.com/channel/UCPdJnvwDFA_7pDeE0H_jr4A https://github.com/orkahub
Yohanes Nuwara	https://github.com/yohanesnuwara
Awesome Reservoir Engineering (Marcelo Albuquerque)	https://github.com/mralbu/awesome-reservoir-engineering
APMonitor	http://apmonitor.com/ https://github.com/APMonitor
Petroleum From Scratch	https://www.youtube.com/channel/UC_lT10npISN5V32HDLAklsw

Get aware: SPE webinars

Data Driven Model for Anomaly Detection – Case Studies from Oil and Gas Applications	• https://webevents.spe.org/products/data-driven-model-for-anomaly-detection-case-studies-from-oil-and-gas-applications
AI/ML Drilling Systems Need Timely Trusted Data to Deliver Trusted Results	• https://webevents.spe.org/products/aiml-drilling-systems-need-timely-trusted-data-to-deliver-trusted-results
Data Science Project from End to End: A Sucker-Rod Pump Example	• https://webevents.spe.org/products/data-science-project-from-end-to-end-a-sucker-rod-pump-example
Smart Fields a data driven approach to making oil fields smart	• https://webevents.spe.org/products/smart-fields-a-data-driven-approach-to-making-oil-fields-smart
Using Machine Learning to optimize Completion design	• https://webevents.spe.org/products/**using-machine-learning-to-optimize-completion-design**
Developing Data Science Specialties Targeting Oil & Gas Industry	• https://webevents.spe.org/products/developing-data-science-specialties-targeting-oil-gas-industry
Application of Petroleum Data Analytics to Upstream Oil and Gas Use Cases	• https://webevents.spe.org/products/application-of-petroleum-data-analytics-to-upstream-oil-and-gas-use-cases

Get aware: SPE webinars

A Foundation for Petroleum Data Analytics	• https://webevents.spe.org/products/a-foundation-for-petroleum-data-analytics
Review of Data Analytics Techniques and Data Management Infrastructure	• https://webevents.spe.org/products/a-review-of-data-analytics-techniques-and-data-management-infrastructure
Machine Learning And Data Analytics In Flow Assurance	• https://webevents.spe.org/products/machine-learning-and-data-analytics-in-flow-assurance
Introduction to Data Analysis	• https://webevents.spe.org/products/introduction-to-data-analysis
Tools of Data Analysis (paid)	• https://webevents.spe.org/products/tools-of-data-analysis
Statistics as a Managerial Tool (paid)	• https://webevents.spe.org/products/statistics-as-a-managerial-tool
Data Analysis in the Real World (paid)	• https://webevents.spe.org/products/data-analysis-in-the-real-world

Get aware: SPE webinars

Whose data is it anyway?	• https://webevents.spe.org/products/whose-data-is-it-anyway
Rock Type Classification Using Machine Learning	• https://webevents.spe.org/products/rock-type-classification-using-machine-learning
From Data to Decisions - The Analytics Lifecycle	• https://webevents.spe.org/products/from-data-to-decisions-the-analytics-lifecycle
Data-Driven Analytics (6 sessions program)	• https://webevents.spe.org/products/data-driven-analytics#tab-product_tab_overview
Transforming E&P Applications through Big Data Analytics	• https://webevents.spe.org/products/transforming-ep-applications-through-big-data-analytics-2
PetroTalk: The Case for Incorporating Data Sciences	• https://webevents.spe.org/products/petrotalk-the-case-for-incorporating-data-sciences

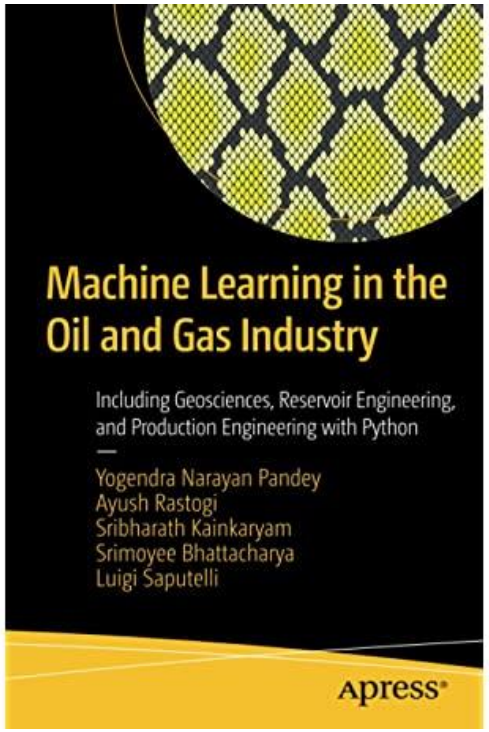
Some Recommended online programs

Stanford University - Machine Learning	• https://www.coursera.org/learn/machine-learning
Python for Data Science and AI	• https://www.coursera.org/learn/python-for-applied-data-science-ai
Master of Science of Machine Learning (Imperial)	• https://www.coursera.org/degrees/msc-machine-learning-imperial
Applied Data Science with Python	• https://www.coursera.org/specializations/data-science-python
The R Programming Environment	• https://www.coursera.org/learn/r-programming-environment
Statistics and Machine Learning	• https://www.coursera.org/specializations/data-science-statistics-machine-learning
Data science courses on edX	• https://www.edx.org/course/subject/data-science
Data science courses on Udemy	• https://www.udemy.com/topic/data-science/
Data Science On DataCamp	• https://www.datacamp.com/
The School Of Data Science in Udacity	• https://www.udacity.com/school-of-data-science
Data Science in KDnuggets	• https://www.kdnuggets.com/
The Open Source Data Science Masters	• http://datasciencemasters.org/

Book Resources

Data Analytics in Reservoir Engineering eBook - June 2020

https://store.spe.org/Data-Analytics-in-Reservoir-Engineering-eBook-P1166.aspx



Machine Learning in the Oil and Gas Industry: Including Geosciences, Reservoir Engineering, and Production Engineering with Python. November 2020

https://www.apress.com/gp/book/9781484260937

https://github.com/Apress/machine-learning-oil-gas-industry

How does data science change the way Engineers and Geoscientists work?

Webinars

Business Area	Today (and yesterday)	What is possible in the immediate future?	What should we target in the mid-term (3+ years)?
Data gathering, validation and analysis	Ad-hoc, manual 1-3 months	**Automated queries, validation and exploratory data analysis**	1000X Faster, access, validation and analysis, robotic insights
Improving reservoir models from data	Leverage available data and experience → lengthy (>6 months)	**Smart data assimilation on model ensemble from geology to planning**	Self-learning models that continuously tune from new data
Knowledge discovery in production data and geologic images	Manual intensive searches (few hours) – SME dependent	**~50% time reduction in few areas (Rock and Borehole images)**	~99% time reduction in most areas (logs, time series, cores, cuttings)
Automating computer-intensive tasks	Use VBA macros and/or commercial platforms	**Use R/Python scripts and/or commercial platforms**	Self-serve from 1000's of community scripts
Production planning and optimization	Manual scenarios generation, Excel intensive, no traceability	**Multiple global optimization scenarios simulation, traceability**	Holistic smarter general AI approach, learned from experience and robotic insights
Opportunity Identification	Ad-hoc, manual 1-3 months	**Continuous, Predictive, Daily**	
Maximize chance of achieving goals	Opportunistic decision making (days-months)	**Holistic smarter general AI approach**	

Conclusions

- Data driven models augment the capabilities of petroleum engineers
- Data Science and Engineering Analytics, part of any personal development and any corporate initiative to digitize operations
- Machine learning models can be developed for any use case meeting the expectations of minimum data availability
- As demonstrated in the case studies, data-driven approaches can generate value
  - 2.5% in net profits by optimizing available resources
  - Reducing the time to perform mundane tasks
  - Maximize chance of achieving goals