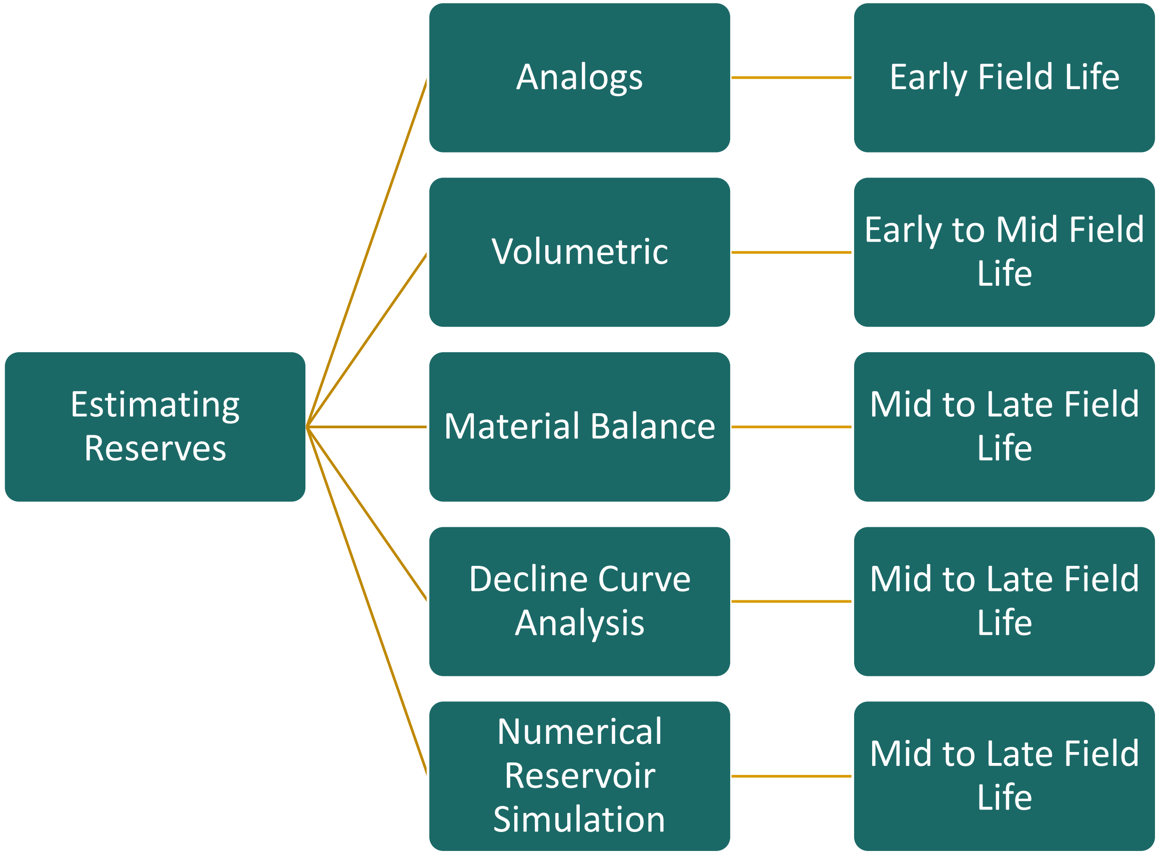
SPE-210769-MS
Applying Data Analytics and Machine Learning Methods for Recovery Factor Prediction and Uncertainty Modelling

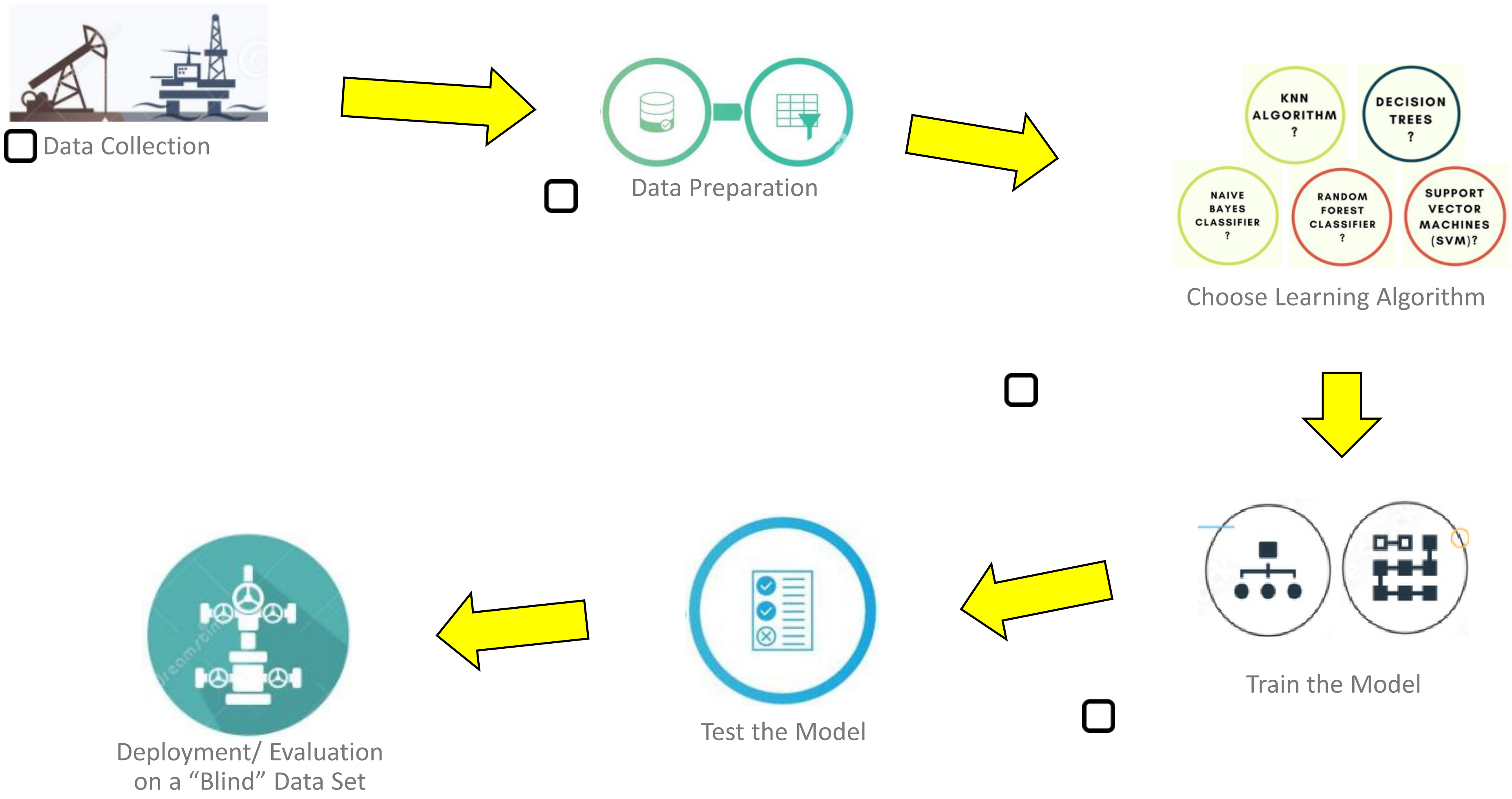
Abel Thomas-Hy, ERCE Australia
Kanna Swaminathan, ERCE Singapore
Munish Kumar, ERCE Singapore
Aizat Rusli, ERCE Malaysia

Motivation

- Recovery Factor (RF) is one the most critical, and sometimes subjective, inputs towards determine field reserve → no clear approach to calculate or estimate given the variety of factors governing its value



Machine Learning Approach



Can We Predict RF?

Develop a workflow to predict RF by applying machine learning (ML) and artificial intelligence (AI) methods

- Use of Geological features
- Use of Categorical features

Apply a "low-code" ML approach that simplifies the ML workflow

Investigate and understand variables that can have an impact on Recovery Factor using ML and AI methods

1. Determine which parameters are of importance "to the machine" in RF prediction
2. Determine and rank the effectiveness of different algorithms
3. Understand some of the ML and AI solutions being developed in literature and industry

Data Sets

Machine learning models developed on two open-source reservoir datasets:

Tertiary Oil Recovery Information System (TORIS) database (1995 Vintage)

- Reservoir database including ~2500 crude oil fields, onshore US
- Representing ~65% of discovered oil onshore united states
- TORIS database focuses primarily on 'numeric' inputs, rather than categoric such as geological feature
- Inherent bias as many onshore US fields have similar development plans of tight well spacing and water injection

Gulf of Mexico (GOM) database

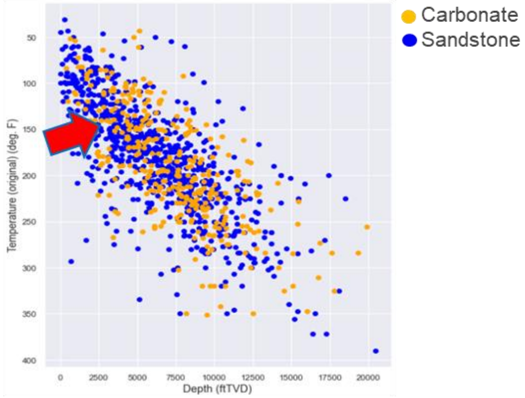
- Collated by Bureau of Ocean Energy Management (US)
- 860 out of 1319 fields abandoned gives good certainty in recorded recovery factor
- RF's based on volumetric and performance based methods for remaining fields
- Dominated by shelf and slope units
- Majority of fields developed with water flooding

Dataset	Categorical columns	Numerical columns	Total Data Set Size	Total Number of Usable Data Points
TORIS	14	56	96,670	9,336
GOM	17	64	1,084,914	59,175

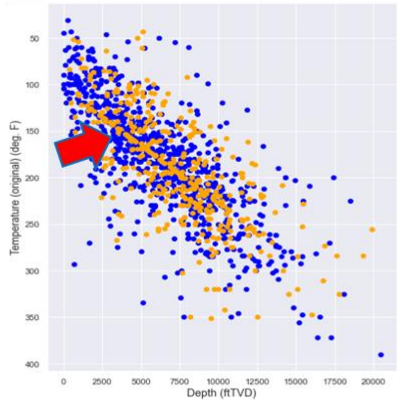
Domain Knowledge

Missing Values

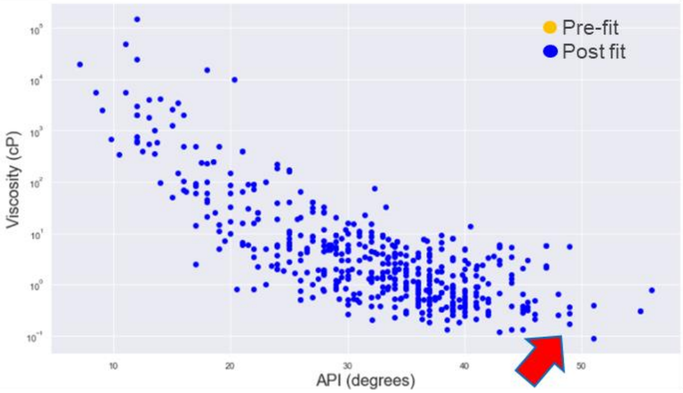
- Temperature-Depth Correlation



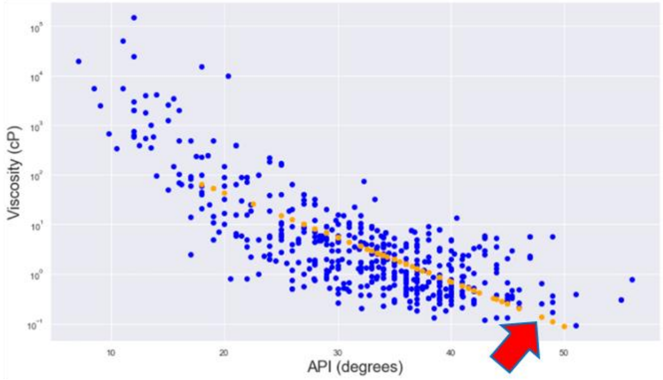
Missing temperature values can be corrected using a "generalised" regression trend.



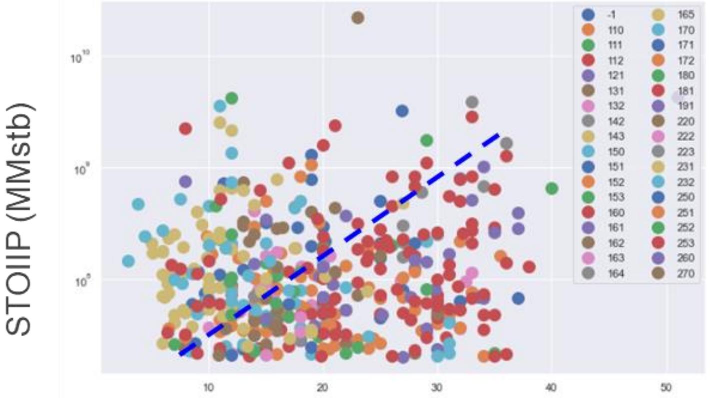
- API-Viscosity Correlation

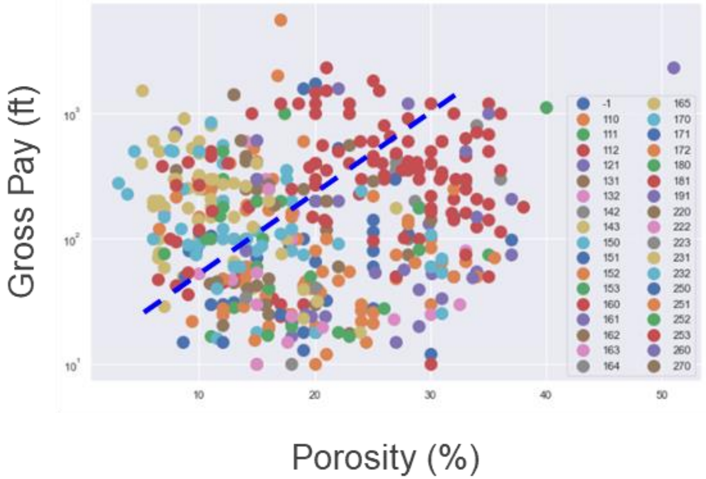


~Missing viscosity values can be corrected for with a correlation trend line between viscosity and API



Correlation ≠ Causality

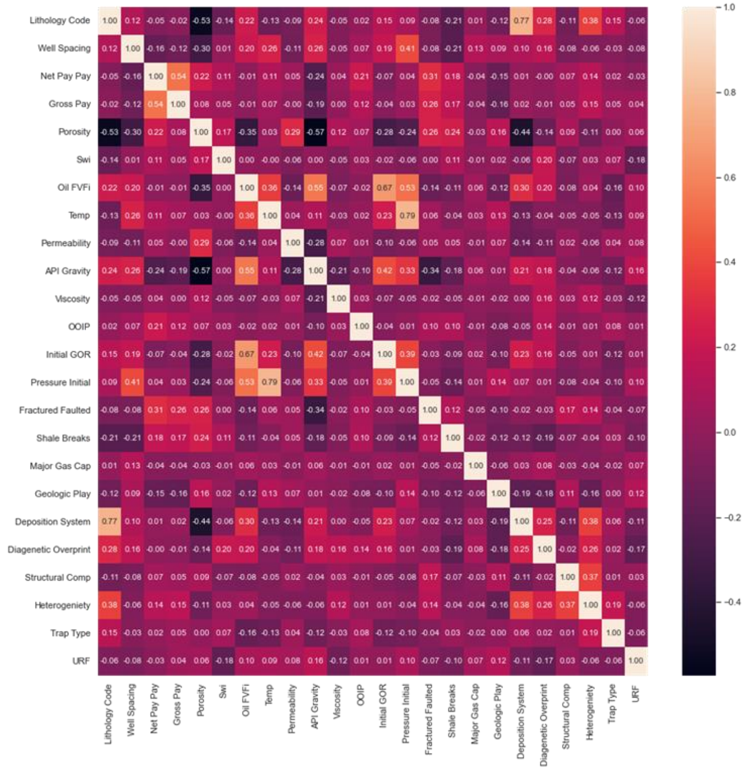


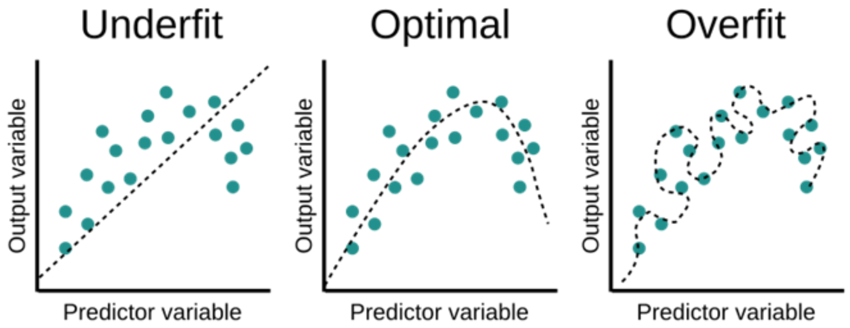


(Multi)collinearity

(Multi)collinearity is a condition where a predictor variable correlates with another predictor variable strongly. Therefore, changes in one of variable would cause changes in the other, giving a highly unstable end model.

- Makes it difficult to determine which variables are significant if the model fluctuates greatly
- Overfits your model because highly correlatable variables DOMINATE the output





Highly correlatable values defined as having coefficients > 0.7.

From the plot, the following 3 parameters show collinearity and do not add additional information to the predictive model.

1. Pressure → Temperature
2. Depositional System → Lithologic Code
3. Initial GOR → Oil FVFi

TORIS Database Inputs

	count	mean	std	min	0.25 percentile	0.5 percentile-	0.75 percentile-	max-
Lithology Code	389	-	-	-	-	-	-	-
Well Spacing	389	36	48	1	10	20	40	640
Net Pay	389	105	169	5	24	50	122	2300
Gross Pay	389	266	356	10	50	150	300	2300
Porosity	389	19	8	3	12	17.6	25	51
Swi	389	31	10	10	25	30	36	68
Oil FVF	389	1	0	1	1.099	1.2	1.33	2.127
Temp	389	138	44	63	105	130	164	266
Permeability	389	401	1507	0.1	10	52	300	26816.5
API Gravity	389	32	9	6	27	34	38	52
Viscosity	389	387	10468	0.07	0.81	2	7	200000
OOIP	389	294	1210	21	48	884	210	22000
Initial GOR	389	516	472	5	200	421	687	4000
Initial Pressure	389	2215	1368	200	1250	1850	2900	9500
Fractured Faulted	389	0	0	0	0	0	1	1
Shale Breaks	389	1	0	0	0	1	1	1
Major Gas Cap	389	3	46	0	0	0	0	680
Geological Play	389	623	713	6	44	414	830	2417
Deposition System	389	184	38	110	152	181	222	270
Diagenetic Overprint	389	2	2	1	1	1	3	9
Structural Complexity	389	14	9	10	10	10	10	50
Heterogeniety	389	1	1	1	1	1	2	3
Trap Type	389	2	1	1	2	2	3	3
URF	389	0	0	0.024	0.25	0.311	0.4	0.5073

TORIS Database:
- 10 Categorical inputs (Lithology code, Fractured faulted etc.)
- 14 Numerical data types (Well spacing, net pay etc.)
- 9,336 datapoints (~10% of original database)

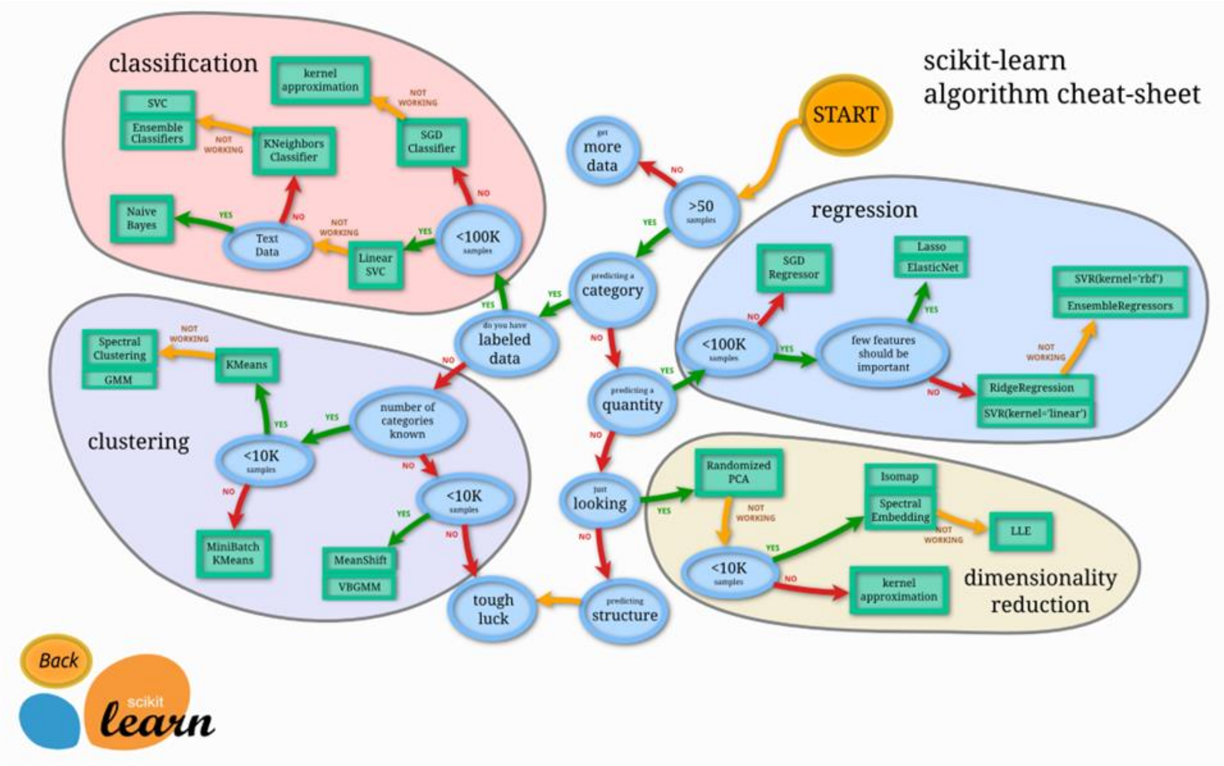
GOM Database:
- 2 Categorical inputs (Chronozone, Drive Mechanism)
- 13 Numerical data types
- 59,175 datapoints (~4% of original database)

How to chose right ML algorithm

Factors to consider:

•Interpretability
•The number of data points and features
•Data format
•Linearity of data
•Training time
•Prediction time
•Memory requirements
→ Not an easy task.

An alternative is to build all and later select the best, but this is time consuming if done "manually".
→ Low Code Machine Learning Libraries



Running the ML algorithms

- We apply a supervised learning approach, splitting the raw inputs into a train-validate-test set (70%-20%-10% split). The "train" set is used to build the model, and the "validate" set is used to test that the model works. We "test" the model using a totally new, never-before-seen data set.
- A total of 20 models were tested as a first pass* using 10-fold cross validation; these models were ranked using the mean absolute error (MAE), the mean squared error (MSE) and the root mean squared error (RMSE).
    - Against outliers, the MSE and RMSE perform better

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - x_i|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - x_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - x_i)^2}$$

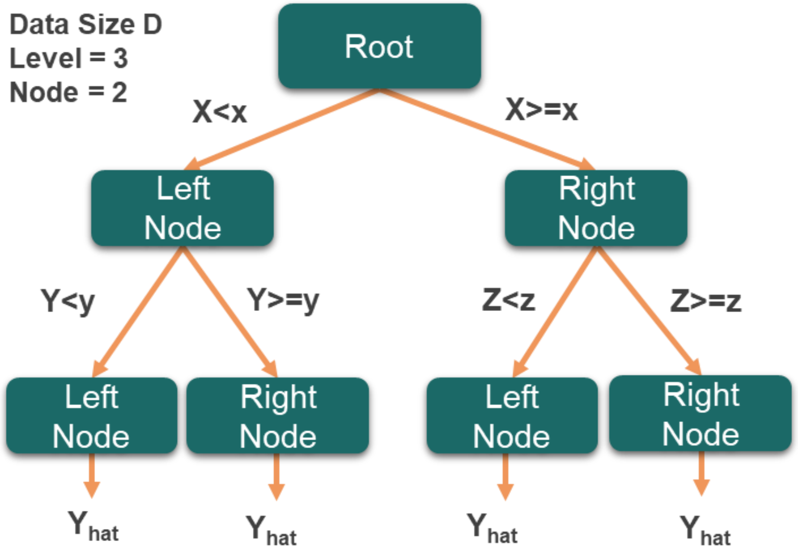
Where y_i is the prediction, x_i is the true value and n is the total number of data points.

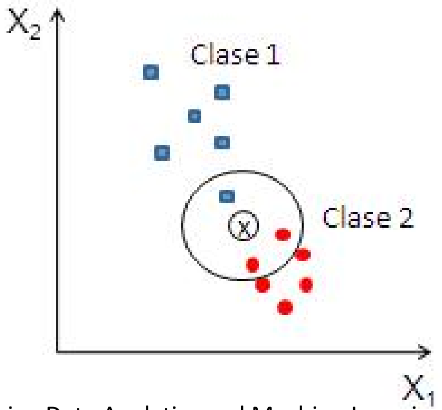
Regressor Model (TORIS)	MAE	MSE	RMSE
Random Forest	0.0791	0.0096	0.0976
Category Boost	0.0821	0.0104	0.1015
K Neighbours	0.0830	0.0109	0.1037

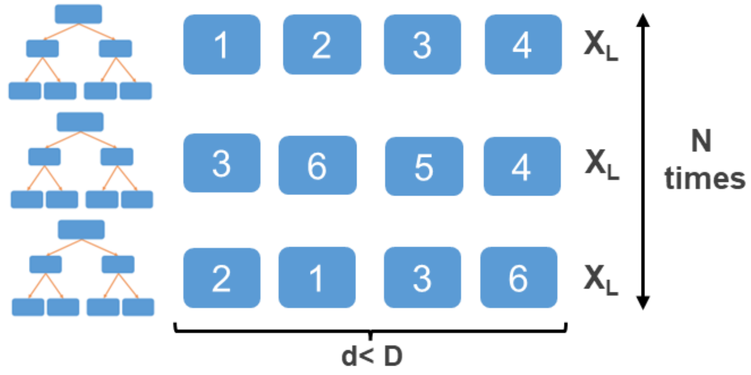
Regressor Model (GOM)	MAE	MSE	RMSE
Random Forest	0.0790	0.0096	0.0978
Category Boost	0.0708	0.0082	0.0906
K Neighbours	0.1091	0.0179	0.1339

Differences in algorithms

- RFR and CatBoost are linked to decision trees (DT)
  - DT can overfit; solution is to use an "bootstrapping" and "ensemble average"
  - CatBoost: Able to handle categorical as well as numeric data. Does this without requiring the conversion of categorical to dummy variables
  - RFR: Insensitive to outliers, works on large number of variables, hard to overfit, requires a lot of CPU memory.
- KNN is a distance-based approach
  - Predicts based on how closely it matches the points of the training set
  - Distance methods – Euclidian, Manhattan (for continuous) and Hamming distance (for categorical).





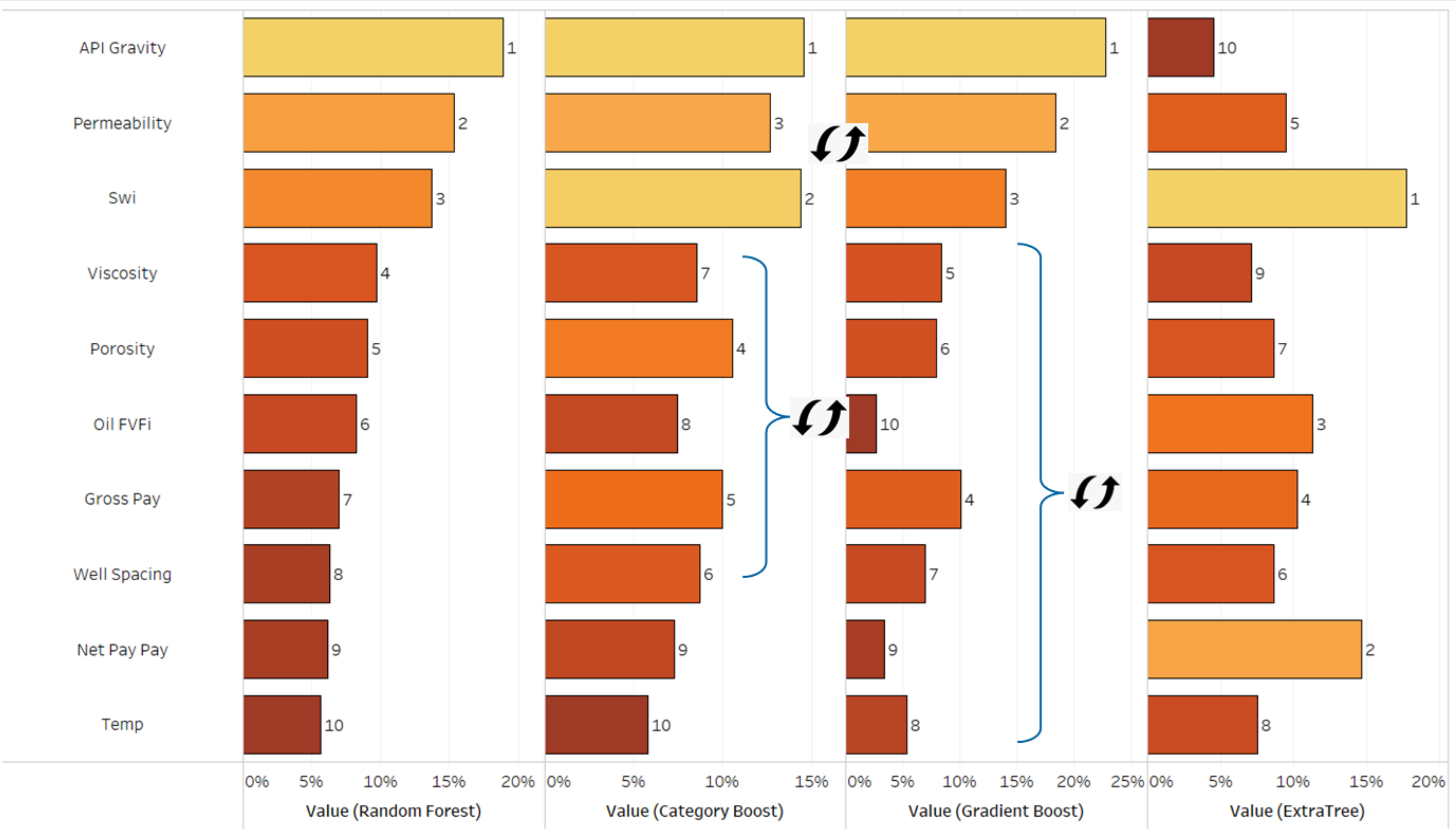


X_L-Sample generated from bootstrap

Top 10 Ranked Features

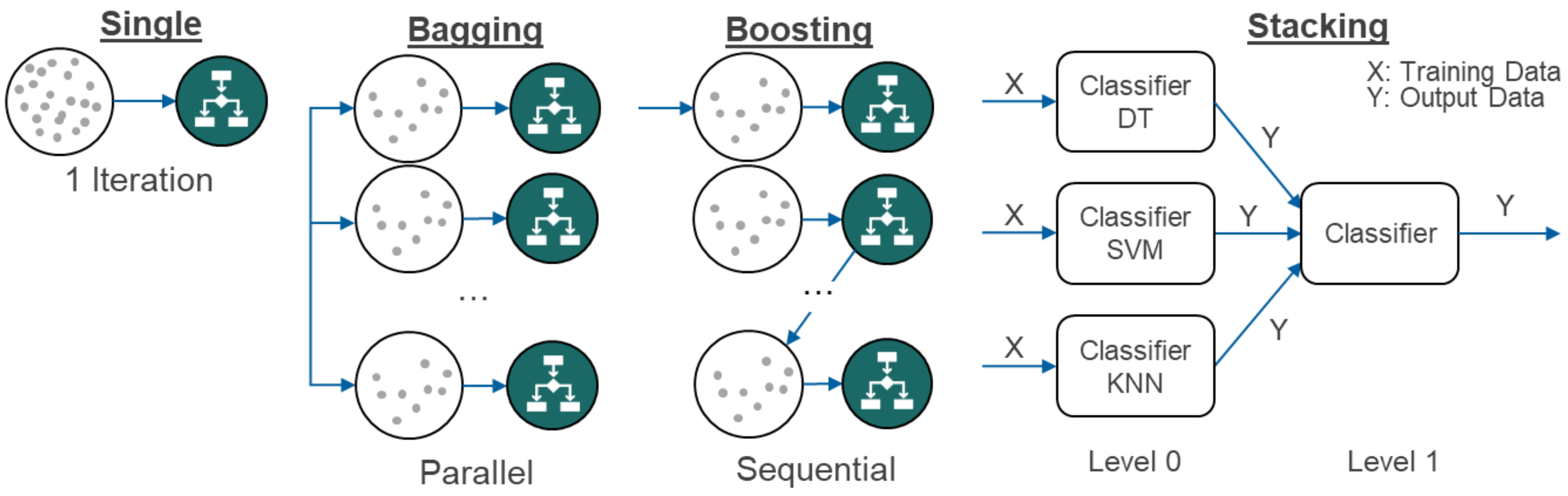
- Some interesting observations for the ranking of the top 10 features:

1) API, permeability, Swi consistently ranked as among the top 3 important variables
2) The remaining variables change in importance ranking depending on the algorithm employed
3) Categorical data has smaller importance, likely due to large degree of categories leading to difficulties in establish trends
4) Extra tree ranking shows a completely different weighting to all the others



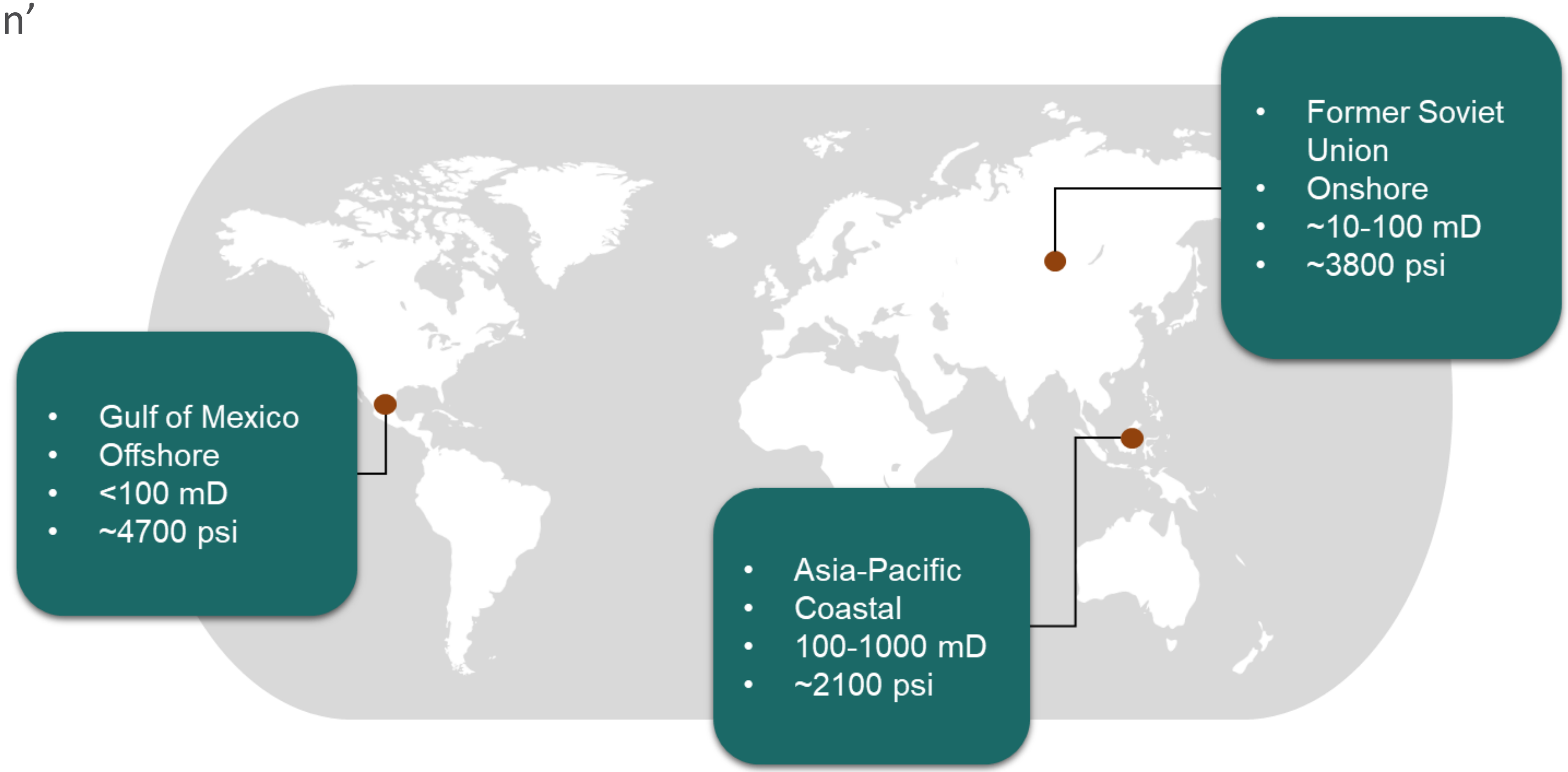
Combining Various ML techniques

- Improving the predictive capability of ML an be done by “averaging” different interpreted models (Ensemble Modelling).
  - **Bootstrap Aggregating (Bagging)** – take (homogeneous) “weak learners” of the same type (same variables each time, but random subset), pass a prediction through each of them and average the result. Each “weak learner” is independent
  - **Boosting** – “Strong learner” = Weighted sum of (homogeneous) “weak learners” with higher importance given to models that were difficult to predict in the previous step.
  - **Blending/ Stacking** – combining different “weak learners” – i.e. multiple models are trained to predict the outcome and a meta-model is created that uses the predictions from those models as an input along with the original features.



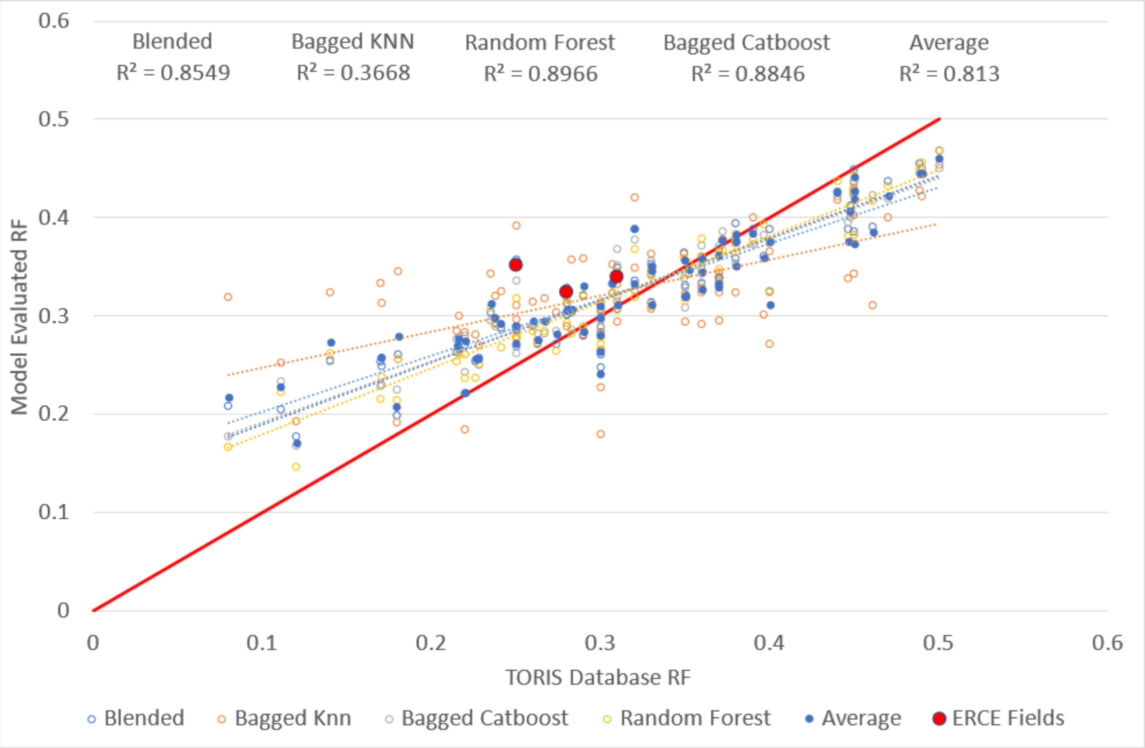
Deployment – Testing on a Blind Data Set

- Machine learning models applied to ‘blind data set’ (hold-out set)
  - Blind dataset is 10% split of data which had been initially separated from training set
- As a double-blind test - ERCE used data from 3 interpreted fields from locations around the world as a test in a ‘real world application’

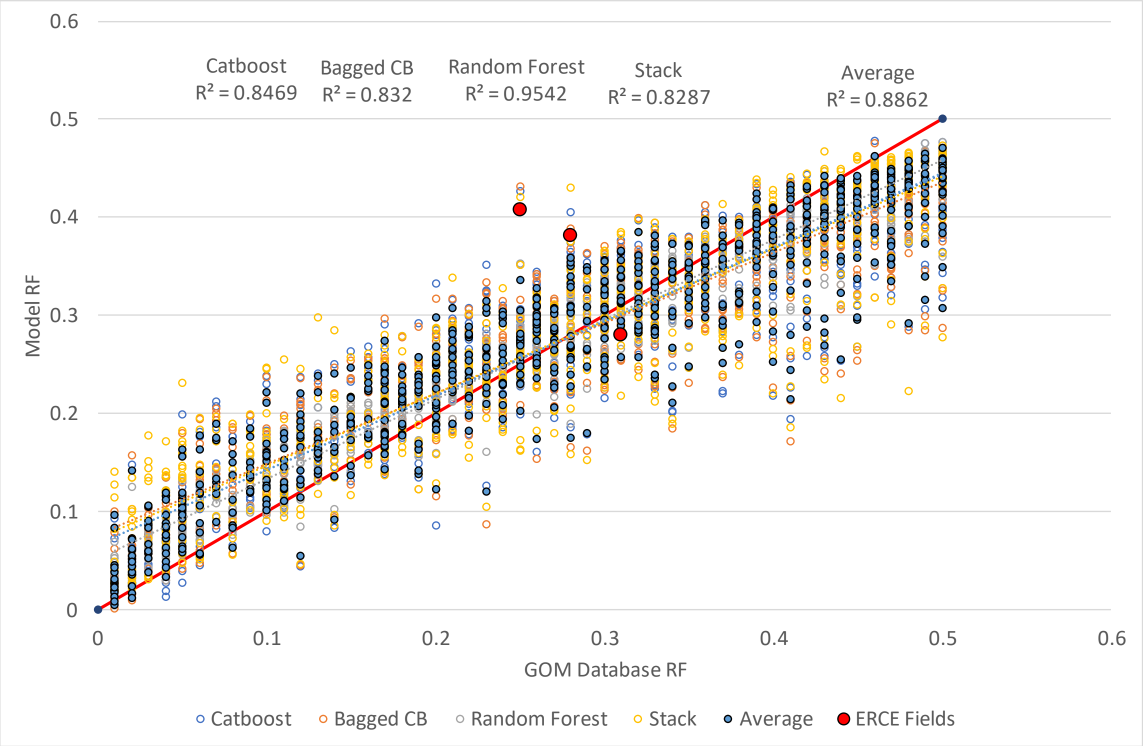


Results

TORIS



GOM



Field	Recovery Factor (V/V)		
	Independent Interpretation	TORIS ML Model	GOM ML Model
Former SU	0.31	0.33	0.28
GOM	0.28	0.32	0.38
Asia Pacific	0.25	0.35	0.40

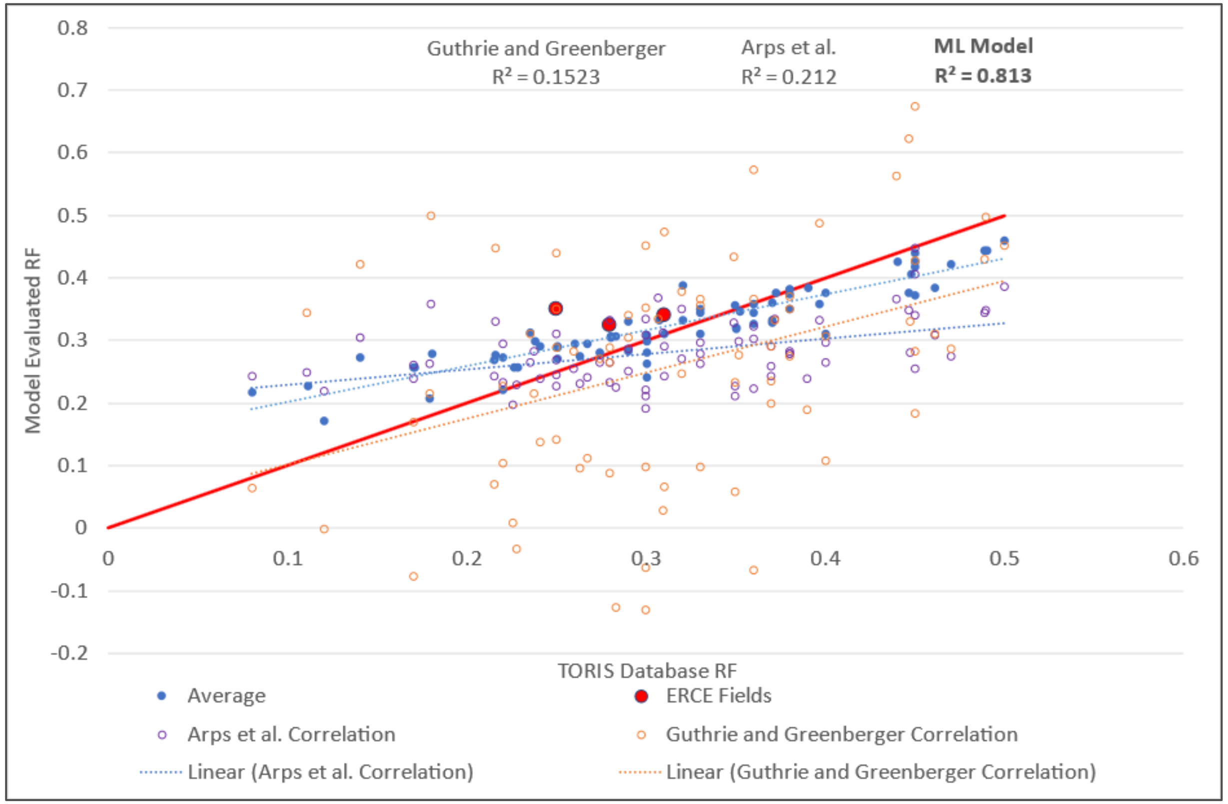
Results – Comparison to conventional correlations

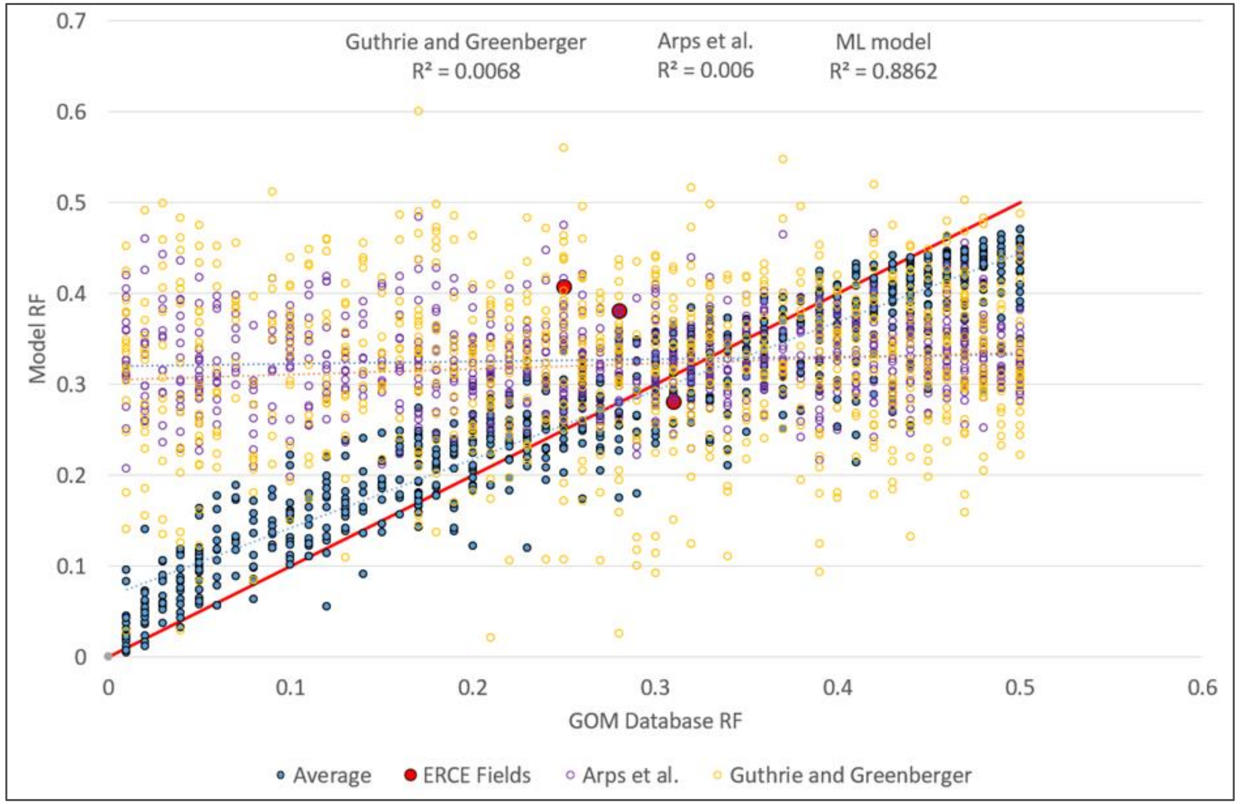
TORIS

R^2 Value		
ML Model	Arps et al.	Guthrie and Greenberger
0.81	0.21	0.15

GOM

R^2 Value		
ML Model	Arps et al.	Guthrie and Greenberger
0.88	0.006	0.007





Conclusions and Learnings

- Models provide basis for recovery factor prediction
- Purpose of using the model must be considered
  - Margin of error (5-10%) shows a model suitable for early life RF estimation – in scenarios where analogues typically used
  - Must be used with caution in marginal fields where error margin can lead to erroneous decision making
- Models are dependent on the quality of the data set
  - Model biased to the location of data set
    - Testing GOM model on fields ERCE evaluated in gulf of Mexico shows good results
    - Testing GOM model on fields in completely different location shows worse results
- Model likely underestimates the effects of ‘categoric’ data compared to numerical data
    - Categoric data separated into too many individual categories – requires mores simplification for trends
    - Resultant likely underestimation of geological setting
- Low code is an effective way to deploy machine learning – but uncertainty in process
  - Low code allows for rapid testing of multiple learning algorithms
  - Machine learning model built separately from low code environment shows similar results for one learning algorithm
- Models show that machine learning can be effectively implemented to predict recovery given sufficient quality data set as long as domain knowledge respected
  - Further work in applying to larger – geographically spread data set
  - Use Neural Networks which are better at non-linearities

Acknowledgements / Thank You / Questions

-