

Well Log Predictor Project

July of 2020

Eloisa Moreira de Lira

Geologist and Reservoir Geophysicist

<https://br.linkedin.com/in/eloisa-lira-a54b13125>

Overview

This project aimed to final submission of Advanced-Data Science Capstone module. This module comprehends the last part of Coursera's Specialization named "Advanced Data Science with IBM" those courses were:

1. Fundamentals of Scalable Data Science
2. Advanced Machine Learning and Signal Processing
3. Applied AI with Deep Learning
4. Advanced-Data Science Capstone

This assignment is destined to conclude the IBM Advanced Data Science Specialization. All these informations came from free datasets referenced below. THIS RESEARCH HAVE ONLY EDUCATIONAL PURPOSE.

Goals

Each item below must be properly covered in the ADD

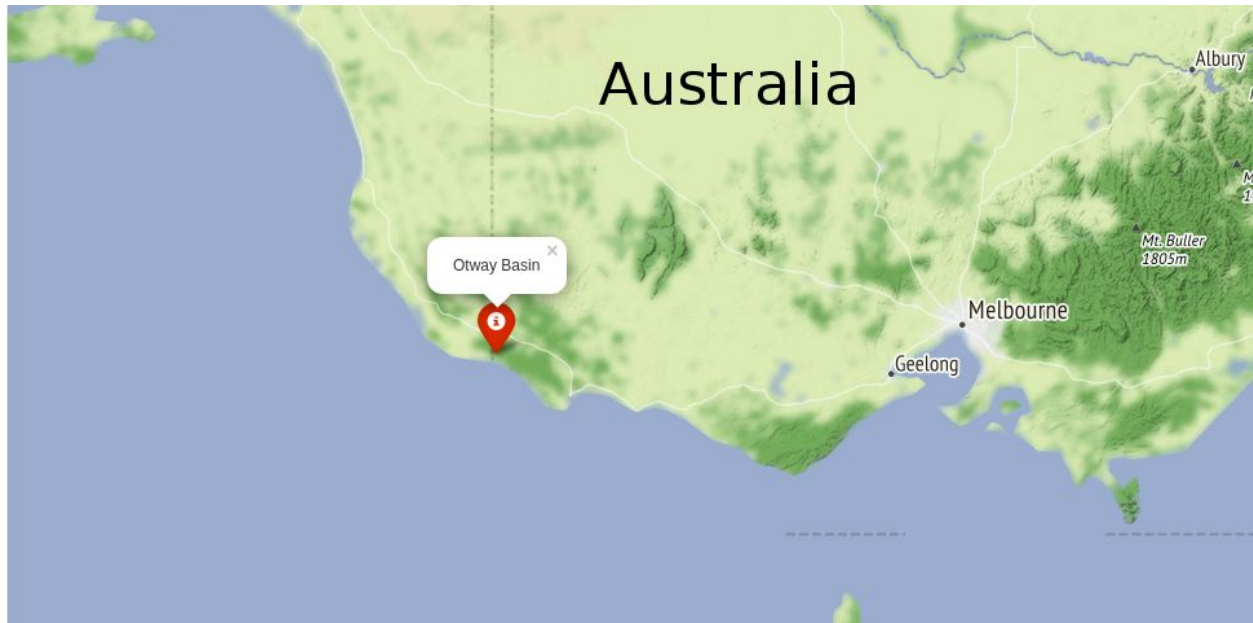
- Why have I chosen a specific method for data quality assessment?
- Why have I chosen a specific method for feature engineering?
- Why have I chosen a specific algorithm?
- Why have I chosen a specific framework?
- Why have I chosen a specific model performance indicator?

Related to notebook activity the following items are properly covered

- The notebooks are clearly structured and documented
- The notebooks contain visualizations in the data quality assessment steps
- The notebooks reveal the overall data processing steps ETL->Feature Engineering->Model Training/Evaluation/Selection->Final Result
- The notebooks contain visualizations in the model performance assessment steps
- The notebooks show the answers to the questions defined in the Use case

Please provide a list of IBM Watson Studio Notebook permalinks GIST to the notebooks you consider as your data product.

Data Case or Geological Overview



Otway Basin Location.

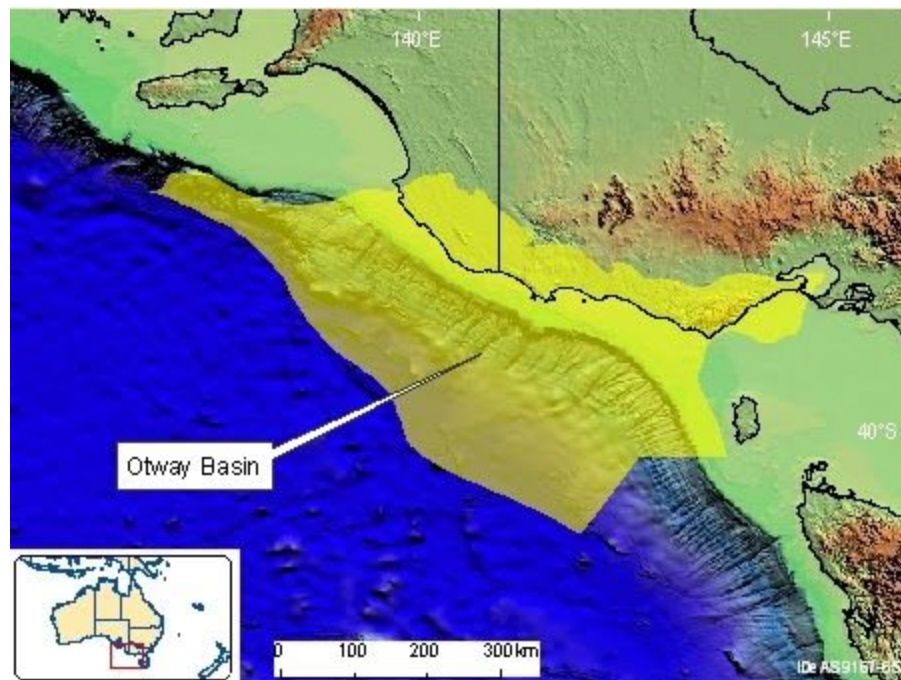
The Otway Basin is north-west to south-east striking, divergent margin, rift, and drift basin. It is approximately 500 km long from Cape Jaffa in South Australia to north-west Tasmania and forms part of the 4 000 km long Jurassic-Cretaceous Australian Southern Rift System.[1]

The basin dates from the Late Jurassic to late Cretaceous periods and formed by multi-stage rifting during the breakup of Gondwana and the separation of the Antarctic and Australian plates. The basin contains a significant amount of natural gas and is a current source of commercial extraction.[5]

Summary

Age	Jurassic - Late Cretaceous
Onshore Exploration Well Density	1 well per 121 km ²
Onshore success ratio	12,5%
Offshore exploration well Density	1 well per 5,000 km ²
Depth to target zone	1000-4000+ m
Thickness	>9 km
First commercial discovery	1987 (Katnook 1) 1967 CO ₂ (Caroline 1)
Identified reserves	78.1 PJ (74 x 10 ¹² Btu) sales gas (in 6 fields)


·Source data[4]



Location of Basin in yellow-greenish.

Here is the brief text that explains the Basin history from reference[1].

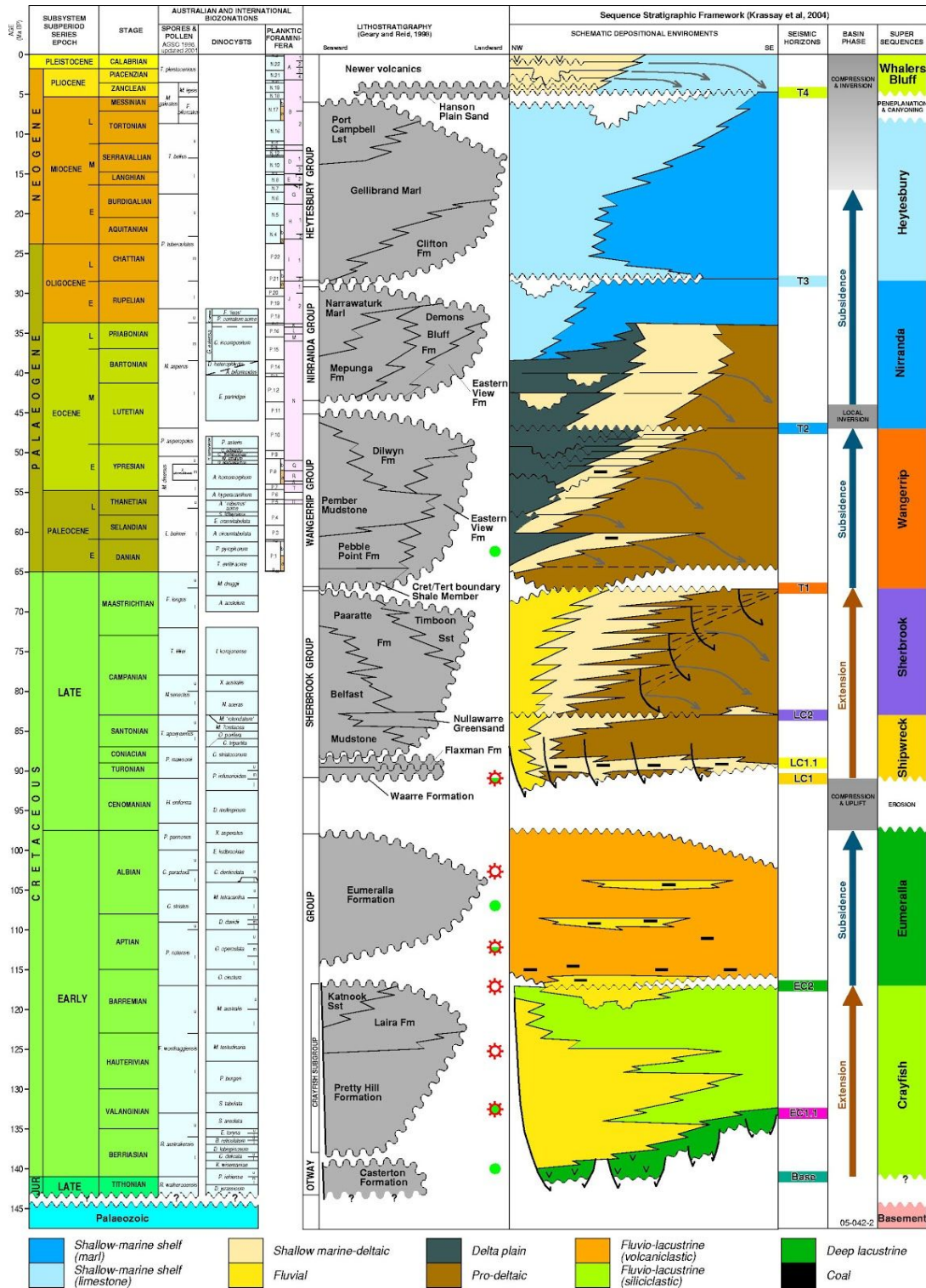
The Late Jurassic-Cenozoic Otway Basin is a large, northwest-trending on/offshore basin on the southern Australian passive margin. Exploration is mature onshore and immature offshore, with >200 wells in South Australia, Victoria, and Tasmania. Commercial gas discoveries include industrial-grade CO₂. No commercial oil discoveries have been identified.



The basin formed by multi-stage rift-sag and inversion phases. Late Jurassic to Early Cretaceous rifting resulted in the east-west trending Inner Otway Basin. Late Cretaceous rifting, culminating in a continental breakup in the Maastrichtian, produced northwest-southeast trending depocentres beneath the outer shelf and slope. Multiple phases of compression in the Cretaceous-Recent resulted in the inversion and wrenching of pre-existing structures.

The basin contains five major depocentres, the mainly onshore Inner Otway Basin, the offshore Morum, Nelson and Hunter Sub-basins, and eastern Torquay Sub-basin. The Latest Jurassic-Early Cretaceous Otway Supergroup comprises up to 8 km of continental and fluvial-lacustrine sediments that accumulated in grabens and half-grabens of the first rifting event. Coastal-plain, deltaic and marine sediments of the Late Cretaceous Sherbrook Group are up to 5 km thick. The Paleocene-middle Eocene Wangerrip Group sediments were deposited in the coastal plain, deltaic, and inner shelf settings and are separated from the open marine, mixed carbonates/siliciclastics of the Eocene-Miocene Nirrandra and Heytesbury groups, by a major unconformity.

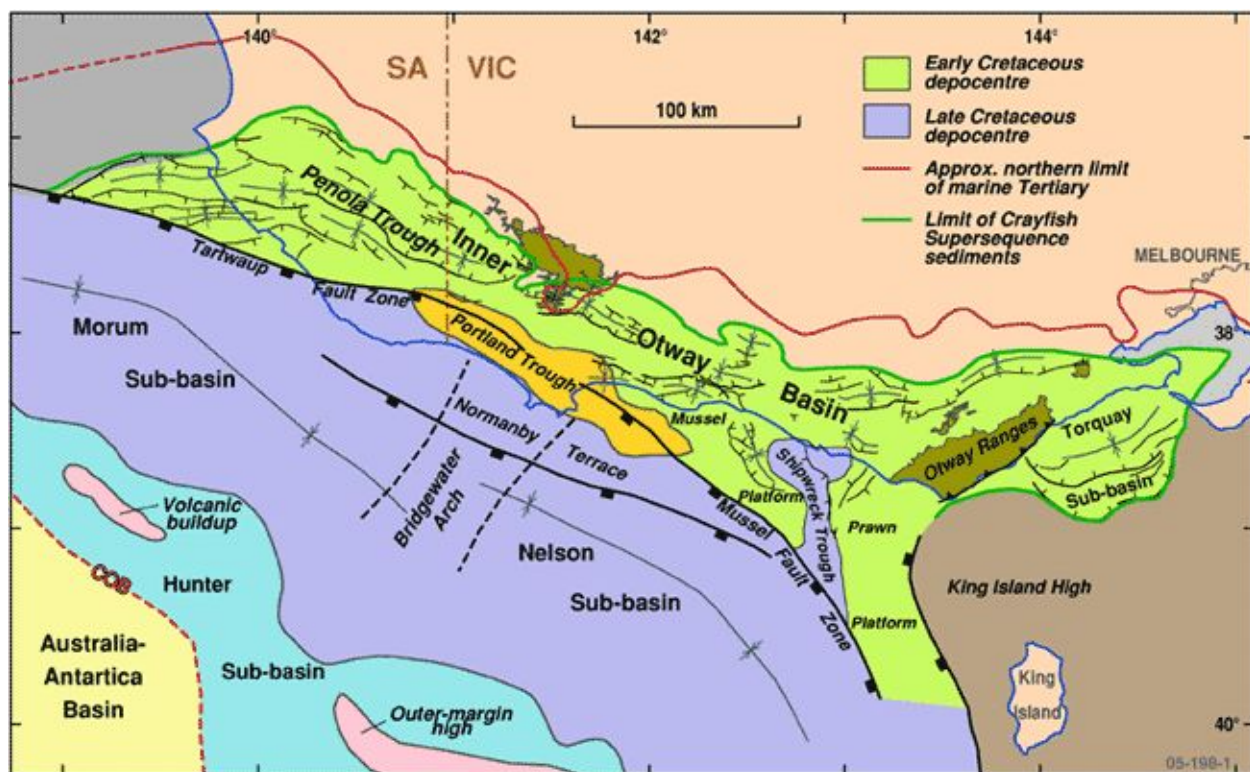
The main exploration targets in the Otway Basin are the Waarre Sandstone at the base of the Sherbrook Group and sandstones of the Pretty Hill Formation and Katnook Sandstone/Windermere Sandstone Member in the Early Cretaceous section. The main source rocks occur in the Early Cretaceous section. Regional and intraformational seals exist in the Pretty Hill, Laira, Eumeralla and Flaxman formations, the Belfast, Skull Creek, and Pember mudstones and mudstones and marls of the Wangerrip, Nirrandra and Heytesbury groups. Play types include faulted anticlines, large anticlinal features, and tilted fault blocks.



Stratigraphic and Basin Event Chart of the Otway Basin

Structural Elements

The basin includes five significant depocenters. The Inner Otway Basin, Torquay Sub-basin, Morum Sub-basin, Nelson Sub-basin, and Hunter Sub-basin formed as a result of two major, basin-wide rifting phases. At the onset of major north-south rifting in the Late Jurassic, several east-west trending extensional depocenters developed in the onshore part of the basin to define the Inner Otway Basin. Renewed rifting in the Late Cretaceous was driven by a change in crustal extension style from north-south to northeast-southwest resulting in the structurally different, northwest-southeast trending Torquay, Morum, Nelson, and Hunter Sub-basins in the mid- and off-shore. Following each rifting episode, the basin underwent compressional phases resulting in the inversion and wrenching of pre-existing structures.[2]



Otway Basin limits and main structural features in NW.

Use Case

In the oil and gas industry, well logging is the most important technology to collect hard data for detailed description, evaluation, and management of oil and gas reservoirs. This kind of data is a precious input to Geoscientists. However, due to drilling conditions, instrument fault, differences in logging conditions, data loss due to improper storage, and the concern of explorations costs, inevitably there will be incomplete logs or even missing logs.

Nowadays many researchers attempt to reveal the non-linear relationships among different good logs to deal with this multivariate task like machine learning models, linear regression, and neural networks.

This project aimed to use deep neural networks in partnership with Machine Learning techniques to predict missing log data as the same method used in stock markets and medicine behavior.

To discover this lack of data it's mister to use Supervisionated Methods represented by Regression algorithms and strategies to predict the Velocity data.

Supervised learning is where you have input and output variable and then use an algorithm to learn the mapping function from the input to go to output. The aim is to implement the mapping function as close as possible to predict the training data through test data. To reach this the algorithm works as a teacher supervising the process.

Data Set



SARIG is a digital platform that delivers statewide geoscientific and spatial data. It is recommended that SARIG is viewed with a larger screen.

SARIG provides global access to South Australia's key geoscience information, mining and exploration project information, and streamlined business interactions.

Access to SARIG is free and no login is required.

Use SARIG to find:

- South Australian mining and petroleum projects
- Geology, geophysics, regolith, paleo drainage, petroleum basins, groundwater and earthquakes
- Mineral, petroleum and geothermal tenement boundaries
- Exploration company details and commodity information
- Infrastructure locations e.g. power stations, substations and transmission lines, water pipelines, roads and railway, ports, and airports
- Data from other government agencies that value-add to mineral and energy resources managed data, e.g. land access layers for parks, Native Title and property boundaries
- Seismic lines and seismic horizon maps

- SEG-Y and HyLoggerTM image and data downloads

Data Quality Assessment

The whole Basin dataset comprehends 80 gas wells but in terms of workability, only 20 oil fields were chosen inside the Basin. Each field only one well was disposed to free download. There were:

- Balnaves 1
- Viewbank 1
- Wynn 2
- McNamara Park 001
- Northumberland 2
- Banyula 1
- Bool Lagoon 1
- Reedy Creek 1
- Patrick 1
- Camelback
- Lake Hawdon 1
- Crankshaft 1
- Laira 1
- Killanoola Southeast 1
- Katnook 4
- Jacaranda Ridge 1
- Patrick 1
- Cowrie 1
- Hollick 1
- Bungaloo 1

All these wells were reviewed in terms of data content because its common to see well without no data. Some examples are the infill wells or water injection wells. Even more

those kinds, the economical limitation represent an important issue in small or less economical projects.

Data Exploration

To understand how is the aspect of .LAS file, above the well Patrick1 from Adelaide Energy Ltd placed in Otway Basin:

```
~Version -----
VERS.      2.0 : CWLS log ASCII Standard -VERSION 2.0
WRAP.      NO : SINGLE LINE PER DEPTH STEP
CREA. 22-Apr-17 8:59 : 11 AM
~Well -----
STRT .M      0.0 : START DEPTH
STOP .M      2997.8604 : STOP DEPTH
STEP .M      0.1524 : STEP VALUE
NULL .      -999.25 : NULL VALUE
DATE .      22-Apr-17 : LAS file Creation Date
WELL .      Patrick 1 : Well Name
UWI .      Patrick 001 : Unique Well Identifier (UWI)
COMP . Adelaide Energy Ltd : Company
STATE.      SA : State
COUNT.     AUSTRALIA : Country
API .      2681 : API Number
LOC .      Otway : Location
GDAT .      GDA94 Zone 54 : Geodetic Datum
LATI .      5866255.2 : Latitude/Northing
LONG .      478047.69 : Longitude/Easting
EKB .      60.8 : KB Elevation
EGL .      56.5 : GL Elevation
```

TDEP . 2998.0 : Total Depth

DATS . 19-Apr-10 : Spud Date

DATR . 18-May-10 : Rig Release Date

RIGN . Hunt Energy Rig 2 : Rig Name

~Curve Information -----

DEPTH.M : Depth

CALI .in : Caliper CAL Edited, patrick_1_dll.dlis

DRHO .g/cm3 : DensityCorr ZCOR Edited, patrick_1_dll.dlis

DT .us/ft : Sonic DT24 Edited, patrick_1_dll.dlis

GR .gAPI : GammaRay GR , patrick_1_dll.dlis

NPHI .dec : Neutron CNC Edited, patrick_1_dll.dlis

PEF .b/e : PhotoelectricF PE Edited, patrick_1_dll.dlis

RDEP .ohmm : DeepRes RD Edited, patrick_1_dll.dlis

RHOB .g/cm3 : Density ZDNC Edited, patrick_1_dll.dlis

RMED .ohmm : MedRes RS Edited, patrick_1_dll.dlis

RMIC .ohmm : MicroRes RMLL Edited, patrick_1_dll.dlis

SP .mV : SponPot SPSBDH Edited, patrick_1_dll.dlis

~Params -----

~Other -----

~ASCII -----

0.00000	-999.25	-999.25	-999.25	-999.25	-999.25	-999.25	-999.25	-999.25
-999.25	-999.25	-999.25						

0.15240	-999.25	-999.25	-999.25	-999.25	-999.25	-999.25	-999.25	-999.25
-999.25								

Patrick1 has the following well logs:

- Depth corresponds the depth of measurement
- CALI as Caliper from
- DRHO like rock Density
- DT like Sonic. The velocity of the compressional waves through rocks formations.
- GR as Gamma Ray related radioactivity which came from rocks.

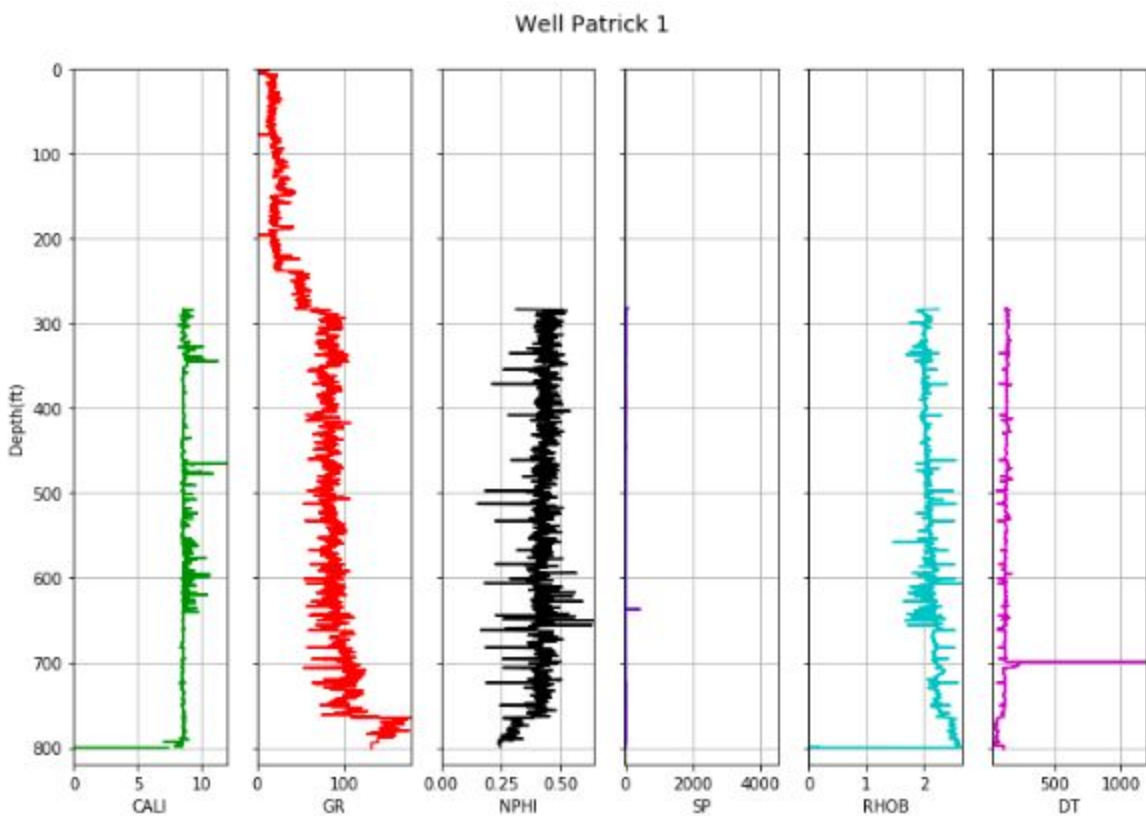
- NPHI as porosity neutron
- PEF as photoelectric Factor
- RDEP as deep rock resistivity
- RHOB like rock density
- RMED as Med Res
- RMIC as Micro Resistivity
- SP corresponds to the spontaneous potential

Another important observation is Null values: in LAS files data are shown by the number -999.25.

Data Visualization

To understand how LAS FILES are plotted let's see an example. Frequently caliper log (CALI at left) shows where the rock was fragile and unreliable. This fact shows where the measurements are susceptible to spurious values caused by those issues during the logging survey. Other points of attention are the initial and final curves they usually present deviated values as well. For those aspects, it's important to proceed a careful data wrangling as presented in the following items.

Bellow is the plotting of the well Patrick1:



```
all_wells.describe() |
```

	CALI	DT	GR	NPHI	RHOB	SP
count	245704.000000	245760.000000	326125.000000	196299.000000	201123.000000	234299.000000
mean	9.361194	96.671125	85.589053	0.247661	2.372112	30.003297
std	1.992433	22.445840	37.548735	0.108318	0.185381	82.302282
min	-3.298500	0.000000	-2.354300	-0.005500	0.000000	-135.375000
25%	8.383600	80.930600	53.953100	0.193100	2.265900	-28.812500
50%	8.798400	91.441600	92.531900	0.255600	2.403900	12.500000
75%	9.789100	108.813100	114.125000	0.315900	2.507600	90.791700
max	23.612400	1183.000000	321.138800	0.928200	3.459000	4528.000000

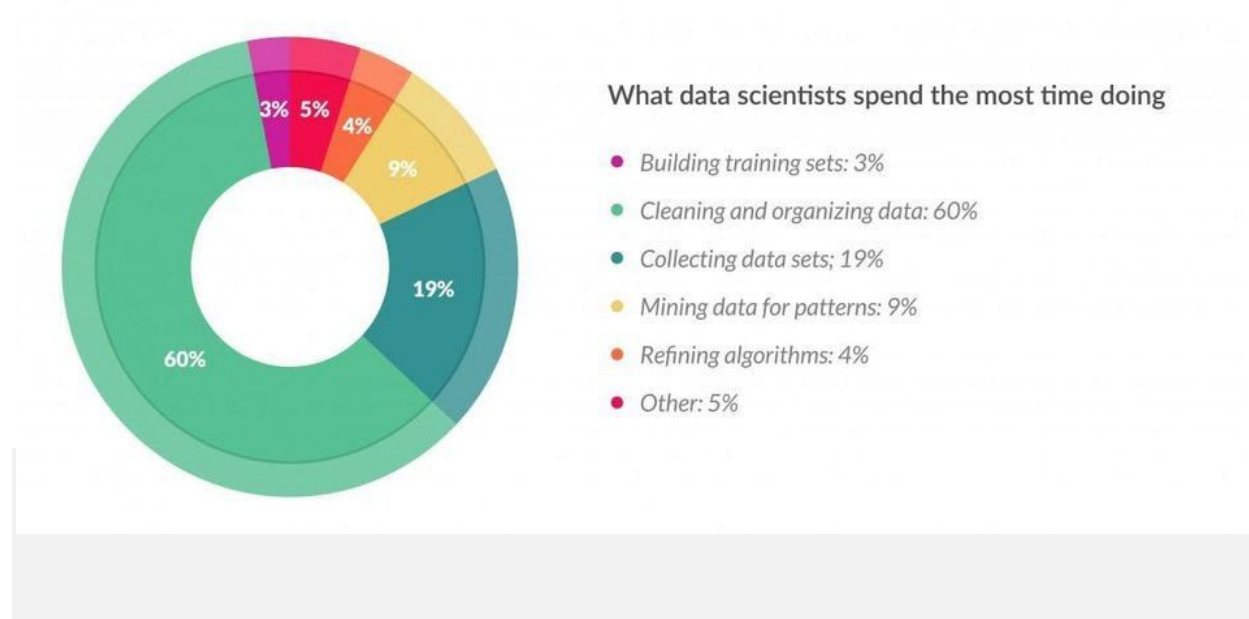
Feature Engineering applied

Feature engineering is the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve the performance of machine learning algorithms. Feature engineering can be considered as applied machine learning itself. [17]

A feature is an attribute or property shared by all of the independent units on which analysis or prediction is to be done. Any attribute could be a feature, as long as it is useful to the model.

The purpose of a feature, other than being an attribute, would be much easier to understand in the context of a problem. A feature is a characteristic that might help when solving the problem.

According to a survey in Forbes, data scientists spend 80% of their time on data preparation:



Source: Forbes [18]

Here are some popular techniques of feature engineering:

- 1.Imputation
- 2.Handling Outliers

3. Binning
4. Log Transform
5. One-Hot Encoding
6. Grouping Operations
7. Feature Split
8. Scaling
9. Extracting Date

Sometimes well logs present values like unusual measurements. To understand what this means the following table shows the normal range for each kind of data:

Log	Min	Max
NPHI	45	-15
GR	0	250
RHOB	1	3
DT	30	140
SP	-10	50

Before to proceed all the data cleaning, its primordial to analyze the whole data. Everything different from that needed is filtered. Next figure represents a brief statistics of each logging:

Seeing in the Notebook:

```
all_wells_dropped= all_wells_dropped[(all_wells_dropped.NPHI > 0) & (all_wells_dropped.NPHI <= 60)]
all_wells_dropped= all_wells_dropped[(all_wells_dropped.SP > -160)&(all_wells_dropped.GR <= 40)]
all_wells_dropped= all_wells_dropped[(all_wells_dropped.GR > 0) & (all_wells_dropped.GR <= 200)]
all_wells_dropped= all_wells_dropped[(all_wells_dropped.RHOB> 1) & (all_wells_dropped.RHOB<= 3)]
all_wells_dropped= all_wells_dropped[(all_wells_dropped.DT > 0) & (all_wells_dropped.DT <= 140)]
```

Missing Values

Very common in LAS files because the depth of interest may vary for specific measurements. Printed values of -999.2500 in LAS files means null value. Missing values are not common in the middle part of the dataset. Usually, it happens in the head or tail of a file. ISNA function returns True if the location has a null value, otherwise, it will be False.

[19]

We may add up these boolean values using sum() function as below:

```
(all_wells.isna().sum())/(all_wells.count())*100
#in_percentage : missing values
```

```
CALI    36.958291
DT      36.927083
GR       3.184975
NPHI    71.428280
RHOB    67.316518
SP      43.625026
```

dropping Rows

This line of code will drop all rows that one (or more) of the subset logs ('GR', 'DT', 'SPOR') has a null value.

```
all_wells_dropped = all_wells.dropna(subset=['CALI', 'DT', 'GR', 'NPHI', 'RHOB', 'SP'],axis=0, how='any')
print(all_wells_dropped.shape)

(191865, 7)
```

Dropped data

```
all_wells_dropped.describe()
```

	CALI	DT	GR	NPHI	RHOB	SP
count	191865.000000	191865.000000	191865.000000	191865.000000	191865.000000	191865.000000
mean	8.840084	91.954288	103.619532	0.249341	2.380250	25.703158
std	1.273156	17.998064	27.052153	0.106976	0.182021	82.941955
min	0.000000	0.000000	0.000000	-0.005500	0.000000	-135.375000
25%	8.326600	79.558300	87.507300	0.195200	2.280400	-26.375000
50%	8.689100	87.727900	104.627400	0.256500	2.411900	6.317700
75%	9.250000	100.484100	122.433600	0.316900	2.511900	74.461500
max	23.612400	1183.000000	261.568600	0.871500	3.192400	4528.000000

Data Rescaling

Many machine learning methods expect or are more effective if the data attributes have the same scale. It's a very important part of the data preprocessing stage. In the Project, we have data with various negative to positive, dozens of units, or simply one or two unities as we will see soon. Before rising the sleeves on that information, it's necessary to rescale all. Two popular data scaling methods are normalization and standardization.[20]

Data Normalization

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.[20]

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Network [21].

In the Project the code stays:


```

: from sklearn.preprocessing import MinMaxScaler
: from sklearn import preprocessing

: # vamos criar um data frame so com as colunas
all_wells_dropped_norm=all_wells_dropped[['DT','GR','NPHI','RHOB','SP']].values
print(all_wells_dropped_norm)

[[ 7.851890e+01  3.863670e+01  9.130000e-02  2.419400e+00 -7.252080e+01]
 [ 8.053390e+01  3.869380e+01  1.143000e-01  2.374900e+00 -7.193750e+01]
 [ 8.107420e+01  3.847380e+01  1.363000e-01  2.346300e+00 -7.154170e+01]
 ...
 [ 7.350590e+01  3.817110e+01  9.900000e-02  2.420900e+00  1.853658e+02]
 [ 7.296150e+01  3.805800e+01  9.690000e-02  2.439500e+00  1.845434e+02]
 [ 7.325270e+01  3.794480e+01  9.480000e-02  2.458800e+00  1.837211e+02]]

: #Let's normalize

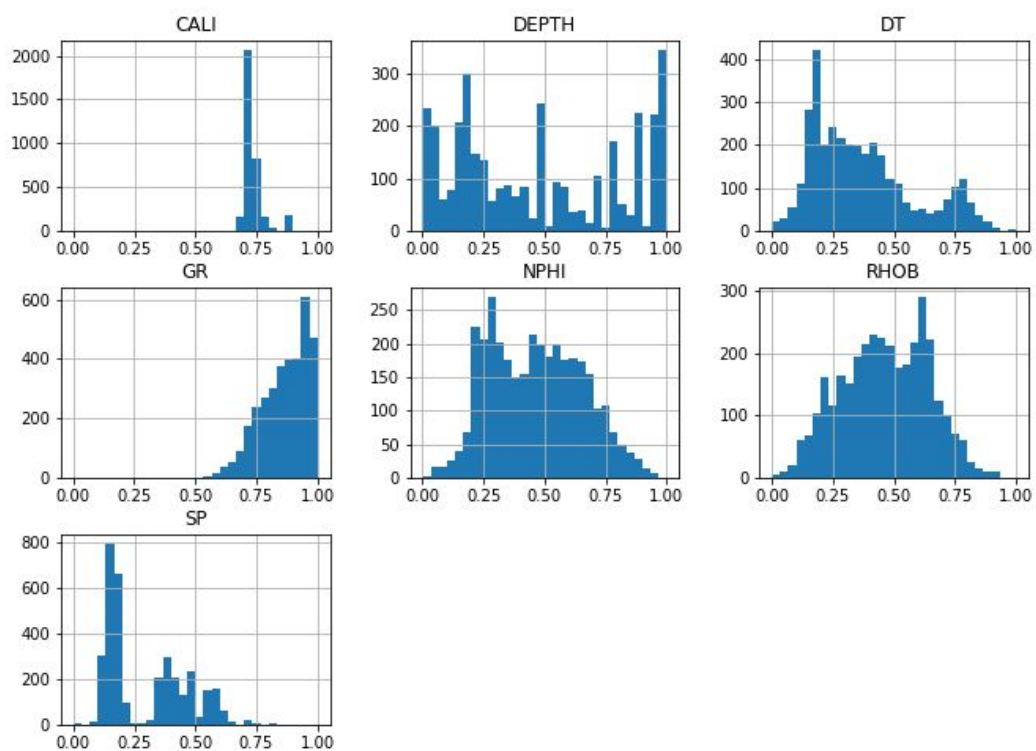
min_max_scaler = preprocessing.MinMaxScaler()
pocos_scaled = min_max_scaler.fit_transform(all_wells_dropped_norm)
all_wells_dropped_normalizado=pd.DataFrame(pocos_scaled)

```

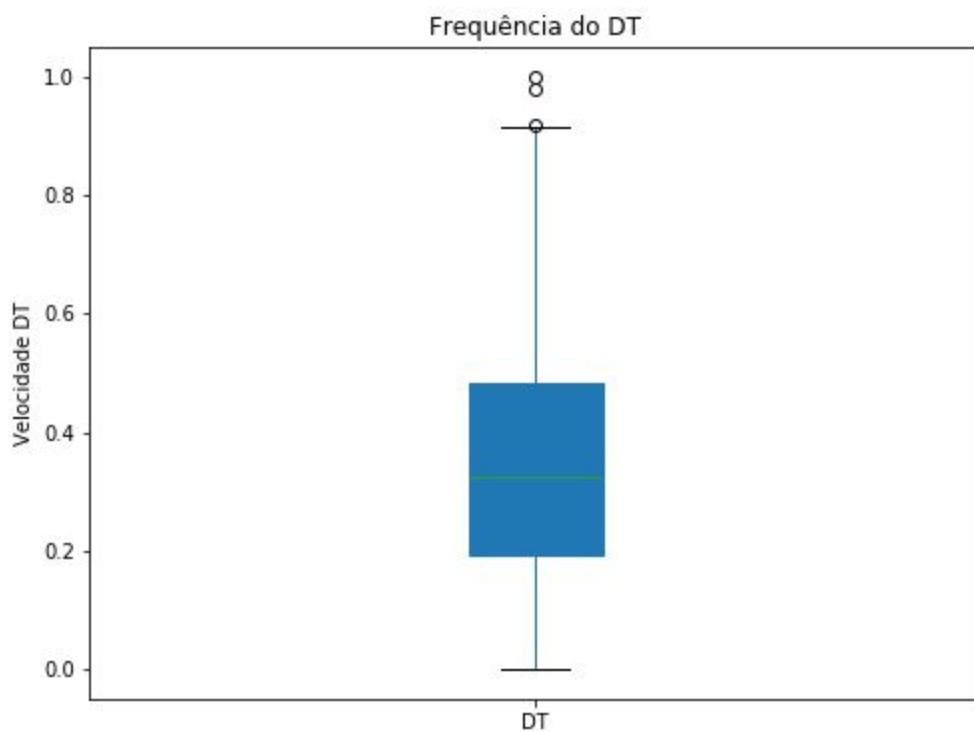
Above the whole data normalized:

	DT	GR	NPHI	RHOB	SP
count	191865.000000	191865.000000	191865.000000	191865.000000	191865.000000
mean	0.077730	0.396147	0.290582	0.745599	0.034541
std	0.015214	0.103423	0.121979	0.057017	0.017786
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.067251	0.334548	0.228848	0.714322	0.023374
50%	0.074157	0.400000	0.298746	0.755513	0.030384
75%	0.084940	0.468075	0.367617	0.786837	0.044997
max	1.000000	1.000000	1.000000	1.000000	1.000000

Distribution of all logs normalization workflow.



Here is the distributions of target log after normalization as well.



One Hot Encode technique

To deal with categorical data 'wells' it's necessary to adopt this tool. It involves representing categorical with a binary vector that has one element for each unique label and marking the class label with a 1 and 0. The main benefit is not weighting a value improperly.

```
#This technique will implemented in Wells column.
```

```
all_wells_dropped_norm_oneHot=all_wells_dropped_normalizado.copy()
```

```
all_wells_dropped_norm_oneHot=pd.get_dummies(all_wells_dropped_norm_oneHot, columns=['Well'])
```

```
print(all_wells_dropped_norm_oneHot.head())
```

	DEPTH	CALI	GR	NPHI	RHOB	SP	DT	\
0	0.494325	0.705311	0.965801	0.214184	0.590563	0.000000	78.5189	
1	0.494381	0.703208	0.967234	0.268920	0.527887	0.002004	80.5339	
2	0.494438	0.701217	0.961715	0.321276	0.487606	0.003363	81.0742	
3	0.494495	0.701268	0.970412	0.364112	0.478169	0.005582	79.7949	
4	0.506583	0.740984	0.996977	0.380533	0.439014	0.014956	81.7513	

	Well_P1	Well_P10	Well_P12	Well_P13	Well_P14	Well_P15	Well_P16	\
0	1	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	
4	1	0	0	0	0	0	0	

	Well_P2	Well_P20	Well_P4	Well_P5	Well_P6
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Selection and justification of Model Performance Indicator

Machine learning is an iterative process. In supervised machine learning, practitioners iteratively collect and label a sample of data, create features to represent the data, train a model with the data and features, and then inspect the model's performance to determine how to proceed in the next iteration (e.g., collecting more data, adding/editing features, experimenting with a different learning algorithm). Once a model has achieved a sufficient level of performance, typically determined by its intended use, the model can be deployed in the target application (e.g., relevance ranking, activity detection, recommendations). [22]

The performance inspection step of the machine learning process can itself be quite involved. Performance inspection typically begins with an assessment of a model's overall ability to correctly predict labels on data, often represented with summary statistics or graphs of common metrics (e.g., accuracy values, precision-recall curves). [22]

Above are the parameters used in this Data Science Project commented one-by-one:

1. Mean Absolute Error

The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. The equation is given in the library references. Expressed in words, the MAE is the average over the verification sample of the absolute values of the differences between forecast and the corresponding observation. The MAE is a linear score which means that all the individual differences are weighted equally in the average. [23]

The diagram illustrates the Mean Absolute Error (MAE) formula with the following components and annotations:

- Formula:** $MAE = \frac{1}{n} \sum |y - \hat{y}|$
- Annotations:**
 - A blue box around $\frac{1}{n}$ is labeled "Divide by the total number of data points".
 - A green box around y is labeled "Actual output value".
 - An orange box around \hat{y} is labeled "Predicted output value".
 - A bracket under the absolute value term $|y - \hat{y}|$ is labeled "The absolute value of the residual".
 - The summation symbol \sum is labeled "Sum of".

2. Mean Squared Error

In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate.[24]

The MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better.

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

where N is the number of data points,
 f_i the value returned by the model and
 y_i the actual value for data point i .

The MSE can be written as the sum of the variance of the estimator and the squared bias of the estimator, providing a useful way to calculate the MSE and implying that in the case of unbiased estimators, the MSE and variance are equivalent [25]

$$MSE(\hat{\theta}) = \text{Var}_{\theta}(\hat{\theta}) + \text{Bias}(\hat{\theta}, \theta)^2.$$

3. R2 Score

The coefficient of determination denoted R^2 or r^2 and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable(s).[26]

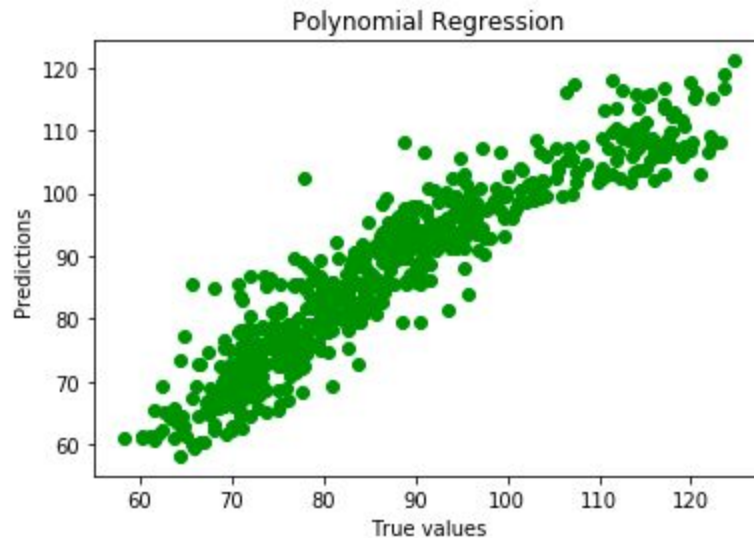
It is a statistic used in the context of statistical models whose main purpose is either the prediction of future outcomes or the testing of hypotheses, on the basis of other related information. It provides a measure of how well-observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.[26]

If \hat{y} is the estimated target output, y the corresponding (correct) target output, and Var is Variance, the square of the standard deviation, then the explained variance is estimated as follow:

$$\text{explainedVariance}(y, \hat{y}) = 1 - \frac{\text{Var}\{y - \hat{y}\}}{\text{Var}\{y\}}$$

The best possible score is 1.0, lower values are worse. For this project the score was:

```
Mean absolute error: 4.08  
R2-score: 0.86  
Residual sum of squares (MSE): 29.02  
Variance score: 0.89
```



At least one traditional Machine Learning Algorithm and one DeepLearning Algorithm applied and demonstrated

Ensemble methods and Machine Learning Model

Ensemble methods aim at improving the predictive performance of a given statistical learning or model fitting technique. The general principle of ensemble methods is to construct a linear combination of some model fitting method, instead of using a fit of the method[27].

Bagging it is a method based on Bootstrap sampling for generating multiple versions of a predictor and using the, in order to get an improved predictor. Usually, the algorithm works for unstable procedures and can push a step toward optimality. But, it can slightly degrade the performance of stable procedures (K. Dana, 2017).[28]

Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model. Base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble. Examples: AdaBoost, Gradient Tree Boosting, ...

The last technique is **stacking**. It comprises several models one after another where the predictions from each model make a new feature.

For this project, the implementations of the last one will help to develop the prediction of accurate log data.

Before starting, It's important to understand that there is no correct way to implement because the whole process is due to combinations in order to get a final generalized model (Wolpert.1991)[30][31].

“Building the Weak Learners”

Weak learner is a learner that no matter what the distribution over the training data is will always do better than chance, when it tries to label the data. The classic weak learner is a decision tree. [30] By changing the maximum depth of the tree, you can control all 3 factors. This makes them incredibly popular for boosting. One simple example is a 1-level decision tree called decision stump applied in bagging or boosting.[31]

Choosing the Learners

- **SVR** Support Vector Regression (SVR) uses the same principle as SVM, but for regression problems.

Those scorings are related to the number of correctly classified samples.

```
svr = SVR()  
svr.fit(X_train, y_train)
```

SVR()

- **XGBRegressor**. XGBoost stands for "Extreme Gradient Boosting" and it is an implementation of gradient boosting trees algorithm. The XGBoost is a

popular supervised machine learning model with characteristics like computation speed, parallelization, and performance.[31]

```
param_dist={}

xgbc=XGBRegressor()
xgbc.fit(X_train,y_train,eval_set=[(X_train, y_train), (X_test, y_test)],eval_metric='logloss',verbose=True)
evals_result = xgbc.evals_result()
```

- **Random Forest Regressor.** A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.[33]

```
: rf=RandomForestRegressor().fit(X_train, y_train)
y_hat_pred=rf.predict(X_test)
```

- **Bagging Regressor.** An ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.[34]

```
: bg=BaggingRegressor()
bg.fit(X_train,y_train)
y_hat_bg=bg.predict(X_test)
```

- **K Neighbors Regressor.** KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses 'feature similarity' to predict the values of any new data points.[35]

```
: knr=KNeighborsRegressor()
knr.fit(X_train,y_train)
y_hat_knr=knr.predict(X_test)
```

- **Decision Tree Regressor.** DTs (Decision Trees) are a non-parametric supervised learning method used for classification and regression. The goal

is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.[36]

```
dtr = DecisionTreeRegressor()
dtr.fit(X_train,y_train)

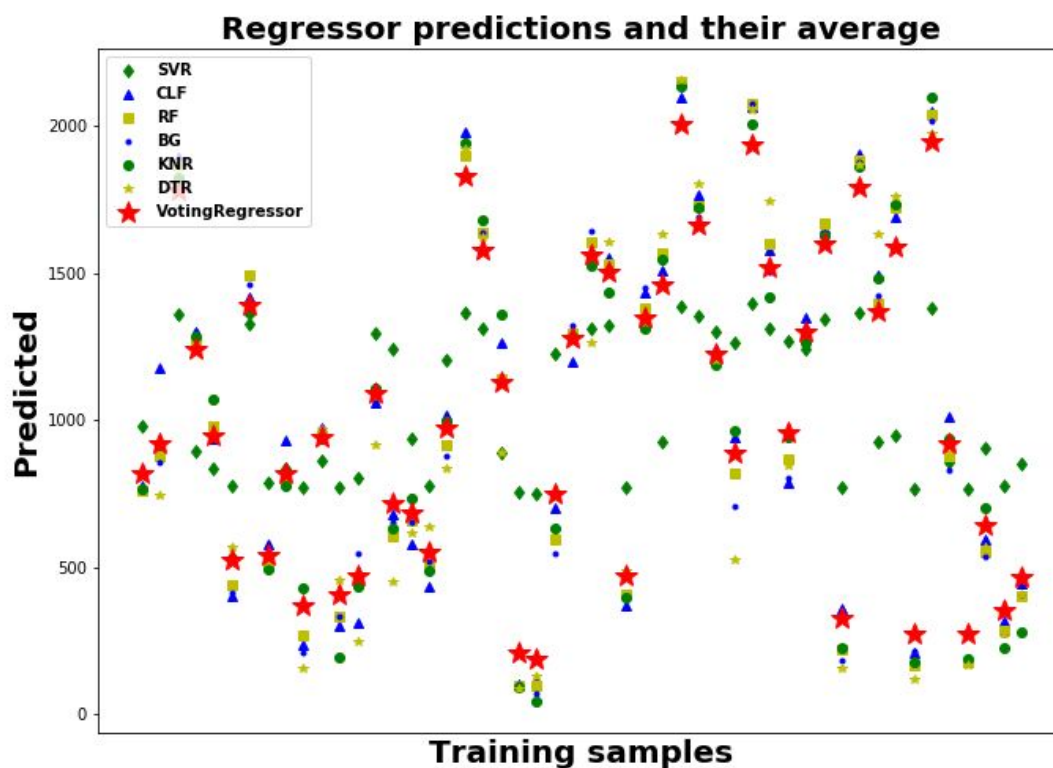
DecisionTreeRegressor()
```

k-Fold Cross-Validation

This technique requires more sophistication on estimative of data. k-Fold Cross validations systematically split up the train data into k-folds, fits the model on k-1 folds, evaluated it on the held-out fold, and repeats this process in each fold. Thus k different models have k several predictions and k different skills scores. Each score is related to k group of samples called folds (for this Project k=3).

```
Evaluate metric(s) by cross-validation and also record fit/score times for each k fold
SVR scores [-285053.7804739 -271777.39365708 -251562.52816508]
clf scores [-23127.62314637 -26889.98805314 -21472.75325947]
rf scores [-20247.38653756 -21432.37727796 -18425.05228555]
bg scores [-21725.19426617 -23796.19331258 -21042.40886675]
knr scores [-22677.7500995 -29581.65135741 -25886.90211706]
dtr scores [-36117.64054726 -42822.97384807 -34854.08094645]
```

To understand which one performed best Sklearn offers the Voting Regressor estimator. Here is the performance of each weak learner:



Implementing the stacking and test the prediction

"Stacked generalization is a method for combining estimators to reduce their biases. More precisely, the predictions of each individual estimator are stacked together and used as input to a final estimator to compute the prediction. This final estimator is trained through cross-validation."

The StackingClassifier and StackingRegressor provide such strategies which can be applied to classification and regression problems." [12]

The architecture of a stacking model involves two or more base models, often referred to as Base models, and a meta-model that combines the predictions of the base models, referred to as a Meta model.[11]

Base-Models: Models fit on the training data and whose predictions are compiled.

Meta-Model: Model that learns how to best combine the predictions of the base models.

The meta-model is trained on the predictions made by base models on out-of-sample data. That is, data not used to train the base models is fed to the base models, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

The outputs from the base models used as input to the meta-model may be real value in the case of regression, and probability values, probability like values, or class labels in the case of classification.

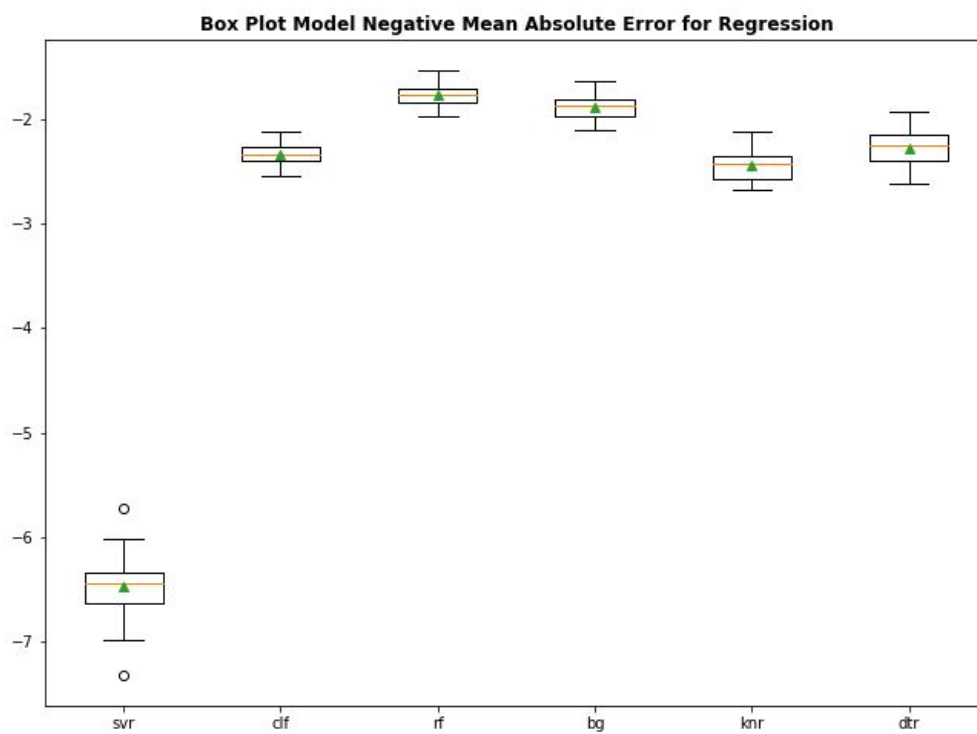
The first score is related to the number of correctly classified samples and the second is R2. Again R2 It provides a measure of how well-observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.[37]

```
def build_model():
    ridge_transformer = Pipeline(steps=[
        ('scaler', StandardScaler()),
        ('poly_feats', PolynomialFeatures()),
        ('ridge', RidgeTransformer())
    ])
    #Level-0 Model (Base-Model)

    pred_union = FeatureUnion(
        transformer_list=[
            ('svr', SVRTransformer()),
            ('clf', GradientBoostingRegressorTransformer()),
            ('bg', BaggingRegressorTransformer()),
            ('rf', RandomForestRegressorTransformer()),
            ('knr', KNeighborsTransformerTransformer()),
            ('dtr', DecisionTreeRegressorTransformer())
        ],
        n_jobs=8
    )

    model = Pipeline(steps=[
        ('pred_union', pred_union),
        ('lin_regr', LinearRegression())])

    return model
```



```

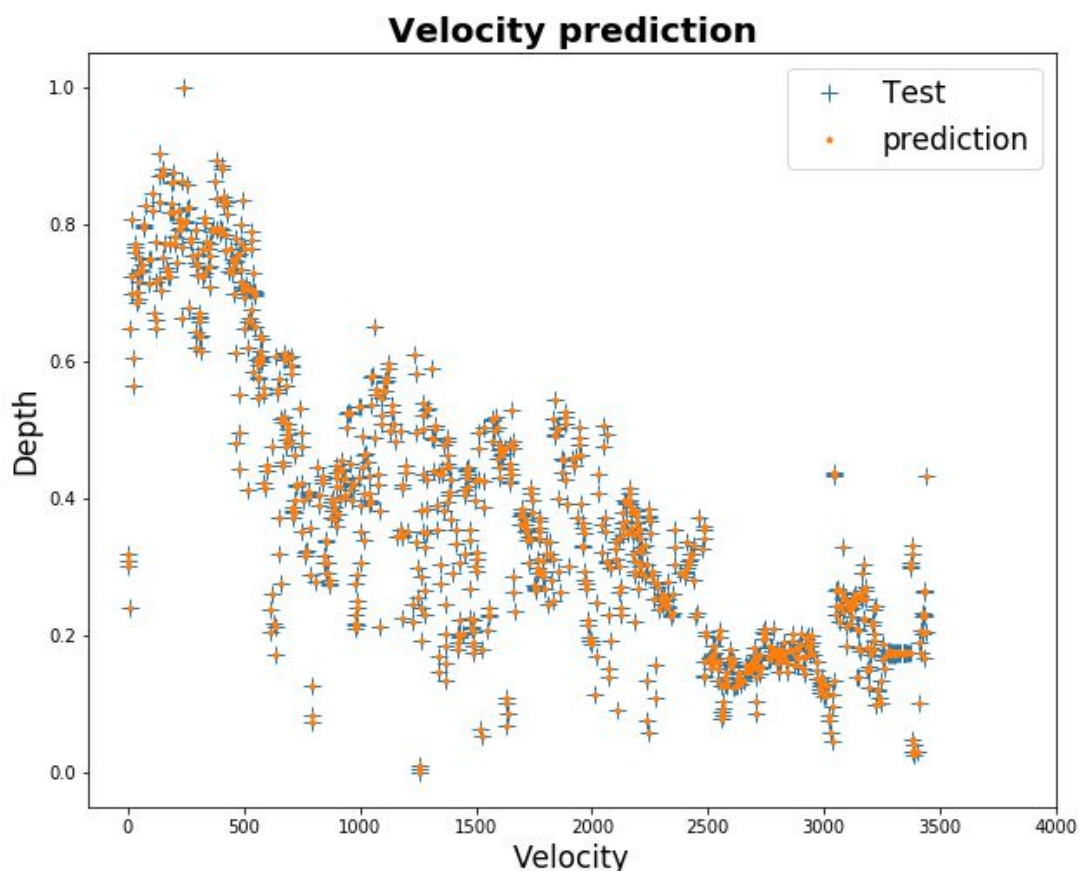
: score_stack_train = ereg.score(X_train, y_train)
  score_stack_test = ereg.score(X_test, y_test)

  y_pred_test=ereg.predict(X_test)
  y_pred_train=ereg.predict(X_train)

  print('Done. R2 Score: {:.2f}'.format( score_stack_train))

```


Done. R2 Score: 0.97



Implementing Deep Neural Net

According to François Chollet in his book **Deep Learning with Python [13]**, deep learning is defined as:

- Deep learning is a specific subfield of machine learning: a new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations.
- How many layers contribute to a model of the data is called the depth of the model.
- Modern deep learning often involves ten or even hundreds of successive layers of representations and they've all learned automatically from exposure to train data.



Meanwhile, other approaches to machine learning tend to focus on learning only one or two layers of representations of the data; hence, there are sometimes called *shallow learning*.

Feedforward neural networks

Reviewing the literature about the use of deep neural networks in geological data recovering the usage of DNN still in development. For those reason this Project tries to reconstruct those cases comparing Ensemble ML and MLN.

Also known as Multi-layered Network of Neurons (MLN). These networks of models are called feedforward because the information only travels forward in the neural network, through the input nodes then through the hidden layers (single or many layers) and finally through the output nodes. In MLN there are no feedback connections such that the output of the network is fed back into itself. [40]

Evaluate Deep Learning models

Let's evaluate using Cross Validation for a multi-layers neural network for the problem. This step is known by Grid Search, it's evaluation of different configurations for DNN (deep neural network) model in order to provide the best estimated performance.[38]

Regression neural networks the metrics adopted was RMSE. Root Mean Squared Error (RMSE) measures the average magnitude of the error by taking the square root of the average of squared differences between prediction and actual observation.[39]

```

: # Function to create model, required for KerasClassifier
def create_model(optimizer='rmsprop', init='glorot_uniform'):

    # create model
    model = Sequential()
    model.add(Dense(9, input_dim=6, kernel_initializer='normal', activation='relu'))
    model.add(Dense(100, kernel_initializer=init, activation='relu'))
    model.add(Dense(1, kernel_initializer=init, activation='linear'))

    # Compile model
    model.compile(loss='mae', optimizer=optimizer, metrics=['acc', 'mean_squared_error'])
    return model

```

Using the function `KerasRegressor` we will pass arguments that will be bundled up and passed on to the fit function called `KerasRegressor`.

```

# create model
model = KerasRegressor(build_fn=create_model, verbose=0)

# define the grid search parameters
optimizers = ['rmsprop', 'adam']
init = ['glorot_uniform', 'normal', 'uniform']

batch_size = [64]
epochs = [100, 500]
param_grid = dict(optimizer=optimizers, batch_size=batch_size, epochs=epochs, init=init)

grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=1, cv=3)
grid_result = grid.fit(X_train2, y_train2)

# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

```

The best result pointed out by grid search method:

Best: -4.821093 using {'batch_size': 64, 'epochs': 500, 'init': 'normal', 'optimizer': 'adam'}

Here are the final DNN configuration. In order to speed up the time consumption the number of batches and epochs was not widely studied.


```
optimizer_final='rmsprop'  
epochs_final=500  
batch_size_final=64
```

```
import keras  
from keras.models import Sequential  
from keras.layers import Dense  
  
model=Sequential()  
model.add(Dense(200,input_dim=6,activation='relu'))  
model.add(Dense(500,activation='relu'))  
model.add(Dense(1,activation='linear'))
```

```
model.summary()
```

```
from keras.optimizers import SGD  
# Compile model  
epochs = 500  
learning_rate = 0.01  
decay_rate = learning_rate / epochs  
momentum = 0.8  
sgd = SGD(lr=learning_rate, momentum=momentum, decay=decay_rate, nesterov=False)
```

```
from keras.utils import to_categorical  
model.compile(optimizer='adam',loss='mae',metrics=['acc','mean_squared_error'])  
y_train2=y_train2.squeeze()
```

Figure below: A plot of loss on the training and validation datasets over training epochs.

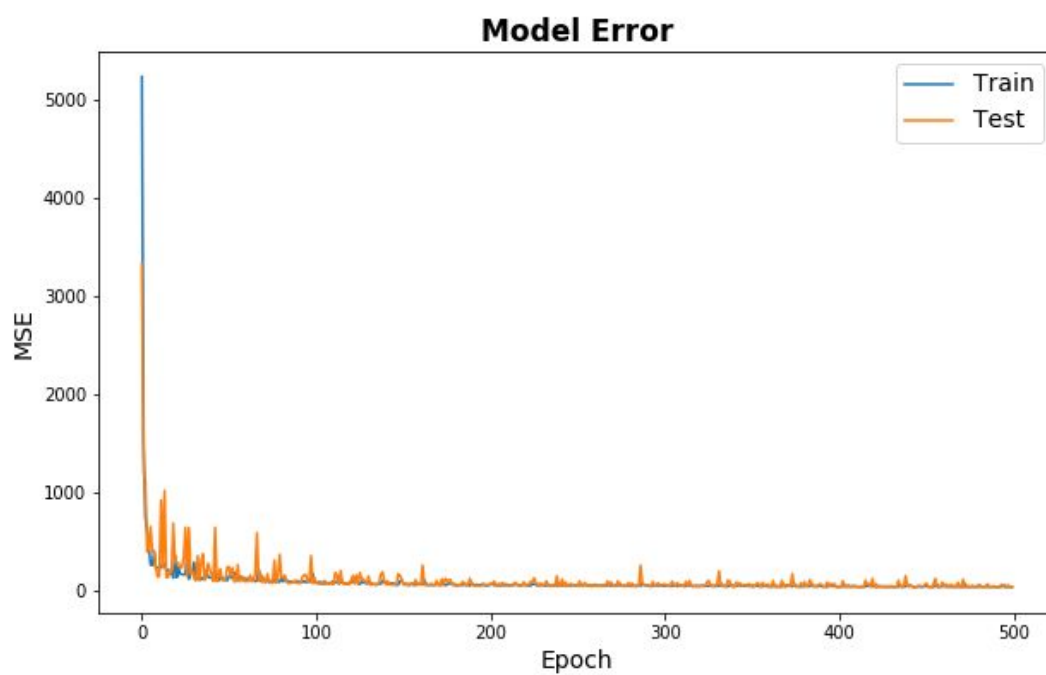
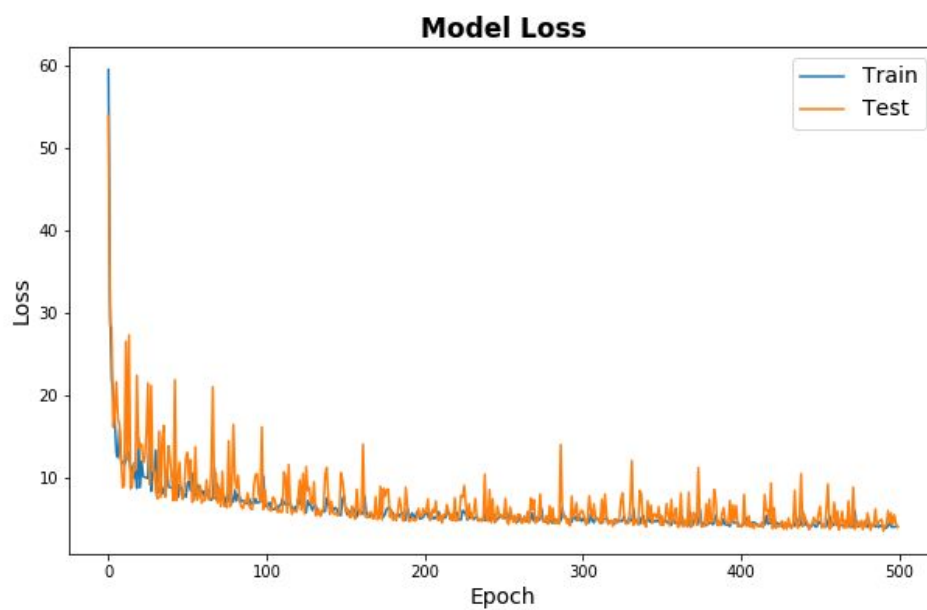


Figure above: A plot of accuracy on the training and validation datasets over training epochs

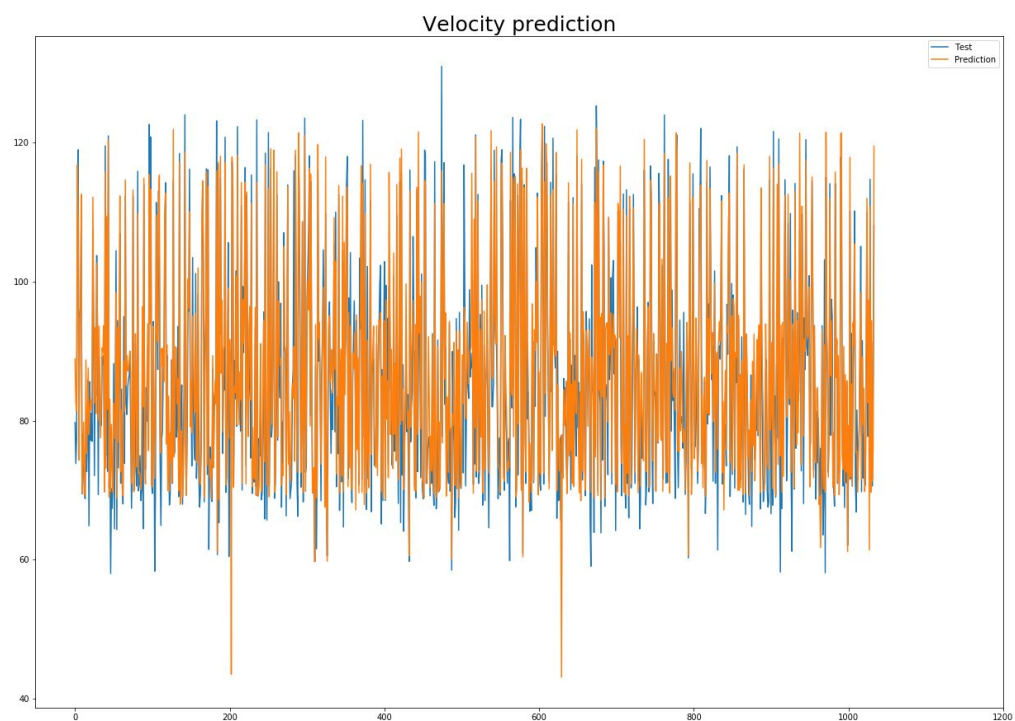
Model prediction on test data

```
pred2=model.predict(X_test2)

score_test=np.sqrt(mean_squared_error(y_test2,pred2))
print("RMSE value on test data:{:.2f}".format(score_test))

RMSE value on test data:5.90
```

Checking the model prediction in relationship with true data.
(the same representation in two ways!)



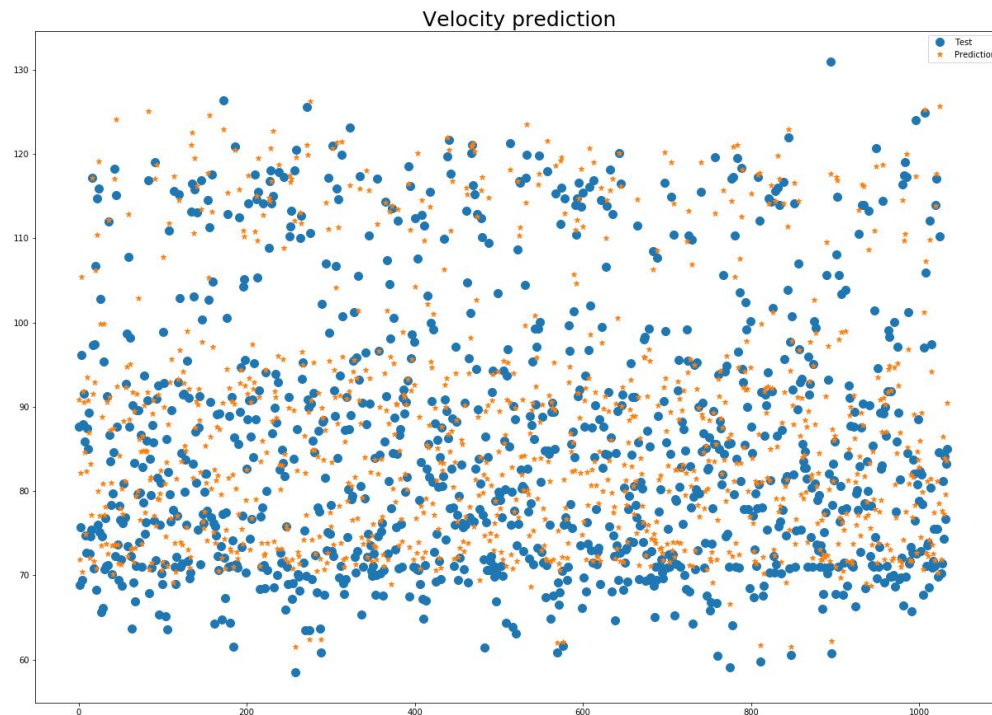


Figure above: in blue is test data and in orange the predicted data.

Model performance between different feature engineerings and models compared and documented

In conclusion, observing the two proposals for predicting well logging data, it is concluded that the Ensemble Machines approach proved to be more efficient.

However, due to the limitation of processing and time consumption during the tests, the neural network suffered penalties from its parameters. However, it is very promising and with excellent generalization capacity.


The tests were performed on the anaconda Framework. Due to the data speed limitation the IIBM Watson environment had to be discarded.


To see whole code:

https://github.com/EloisaLira/Forum_Geologia_completo/blob/master/Projeto_Otway_Basin_DNN_ML_jul2020%20final.ipynb

References

- [1]<https://www.ga.gov.au/scientific-topics/energy/province-sedimentary-basin-geology/petroleum/offshore-southern-australia/otway> , visited in Jun/2020.
- [2]https://en.wikipedia.org/wiki/Otway_Basin, visited in Jun/2020.
- [3]<https://www.spec2000.net>, visited in Jun/2020.
- [4]http://www.energymining.sa.gov.au/petroleum/prospectivity/otway_basin, visited in Jun/2020.
- [5]<https://readthedocs.org/projects/lasio/downloads/pdf/latest/>
- [6]https://wiki.aapg.org/Well_log_analysis_for_reservoir_characterization
- [7]<https://www.statisticshowto.com/absolute-error/>
- [8]<https://www.kaggle.com/stieranka/simple-linear-regression>
- [9]https://en.wikipedia.org/wiki/Mean_squared_error
- [10]<https://machinelearningmastery.com/k-fold-cross-validation/>

- 
- [11]<https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/>
 - [12]<https://scikit-learn.org/stable/modules/ensemble.html>
 - [13]F. Chollet Deep Learning with Python, Ed Manning, (November 2017)
 - [14]<https://towardsdatascience.com/ultimate-guide-to-input-shape-and-model-complexity-in-neural-networks-ae665c728f4b>
 - [15]<https://en.wikipedia.org/wiki/Keras>
 - [16]<https://heartbeat.fritz.ai/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f>
 - [17]https://en.wikipedia.org/wiki/Feature_engineering#cite_note-1
 - [18]<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/>
 - [19]<https://towardsdatascience.com/10-steps-in-pandas-to-process-las-file-and-plot-610732093338>
 - [20]<https://machinelearningmastery.com/rescaling-data-for-machine-learning-in-python-with-scikit-learn/>
 - [21]<https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/>
 - [22] Amershi, S., Chickering, M. Drucker, S. M., Lee, B., Simard, P., and Suh, J. `ModelTracker?Redesigning Performance Analysis Tools for Machine Learning`. In Proceedings of the Conference on Human Factors in Computing Systems. New York. ACM, 2015.
 - [23]http://www.eumetrain.org/data/4/451/english/msg/ver_cont_var/uos3/uos3_ko1.htm
 - [24]https://en.wikipedia.org/wiki/Mean_squared_error#cite_note-pointEstimation-1
 - [25]Wackerly, Dennis; Mendenhall, William; Scheaffer, Richard L. (2008). Mathematical Statistics with Applications (7 ed.). Belmont, CA, USA: Thomson Higher Education. ISBN 978-0-495-38508-0.
 - [26]<https://brainly.in/question/8454152>
 - [27] Bühlmann, Peter, 2004. "Bagging, boosting and ensemble methods," Papers 2004,31, Humboldt University of Berlin, Center for Applied Statistics and Economics (CASE).
 - [28]<https://bradleyboehmke.github.io/HOML/bagging.html>
 - [29]Wolpert, D. H. (1992). Stacked generalization. Neural Networks, 5(2), 241–259. doi:10.1016/s0893-6080(05)80023-1

- 
- [30]<https://stats.stackexchange.com/questions/82049/what-is-meant-by-weak-learner>
- [31]Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10), 993–1001. doi:10.1109/34.58871
- [32]<https://www.datatechnotes.com/2019/06/regression-example-with-xgbregressor-in.html>
- [33]<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>
- [34]<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html>
- [35]<https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/>
- [36][http://scikit-learn.org/stable/modules/tree.html#:~:text=Decision%20Trees%20\(DTs\)%20are%20a,inferred%20from%20the%20data%20features.](http://scikit-learn.org/stable/modules/tree.html#:~:text=Decision%20Trees%20(DTs)%20are%20a,inferred%20from%20the%20data%20features.)
- [37]https://en.wikipedia.org/wiki/Coefficient_of_determination
- [38]<https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/>
- [39]Fadi Al-Turjman, The Cloud in IoT-enabled Spaces, CRC Press. 2020