Aramco Upstream Solutions Technathon

26-28.10.18

Challenge #1

Reservoir Evaluation

Oleg Cheremisin
(mktoid)

Pure ML solution

Let's assume that we know nothing about underlying physics and learn from the data using quite powerful machine learning model that can deal with non-linearities.

So if we have data prepared using the same process,
we can train the model on dataset with known target
(Fluid_Fraction)
and then make a prediction for other dataset

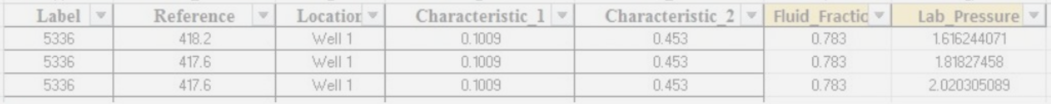
Technathon

Data Structure

1 Clustering

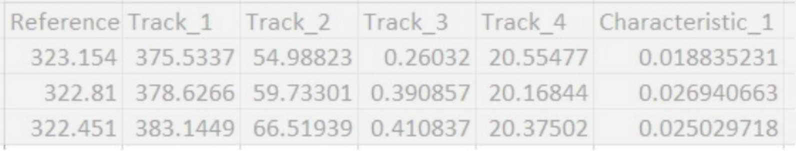
- Lab_Curves
- Logs-5
- Logs-4

Lab_Curves



Label	Reference	Locatior	Characteristic_1	Characteristic_2	Fluid_Fractic	Lab_Pressure
5336	418.2	Well 1	0.1009	0.453	0.783	1.616244071
5336	417.6	Well 1	0.1009	0.453	0.783	1.81827458
5336	417.6	Well 1	0.1009	0.453	0.783	2.020305089

Logs

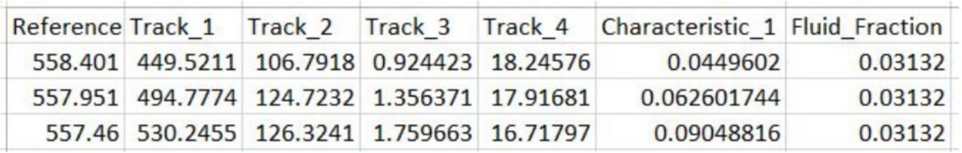


Reference	Track_1	Track_2	Track_3	Track_4	Characteristic_1
323.154	375.5337	54.98823	0.26032	20.55477	0.018835231
322.81	378.6266	59.73301	0.390857	20.16844	0.026940663
322.451	383.1449	66.51939	0.410837	20.37502	0.025029718

2 Training

- Logs-B
- Logs-C
- Logs-D_Lost_Sections

Logs

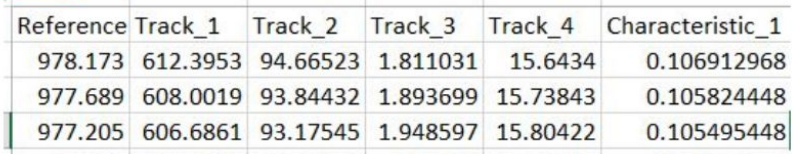


Reference	Track_1	Track_2	Track_3	Track_4	Characteristic_1	Fluid_Fraction
558.401	449.5211	106.7918	0.924423	18.24576	0.0449602	0.03132
557.951	494.7774	124.7232	1.356371	17.91681	0.062601744	0.03132
557.46	530.2455	126.3241	1.759663	16.71797	0.09048816	0.03132

3 Testing

- Logs-A
- Logs-D_Lost_Sections

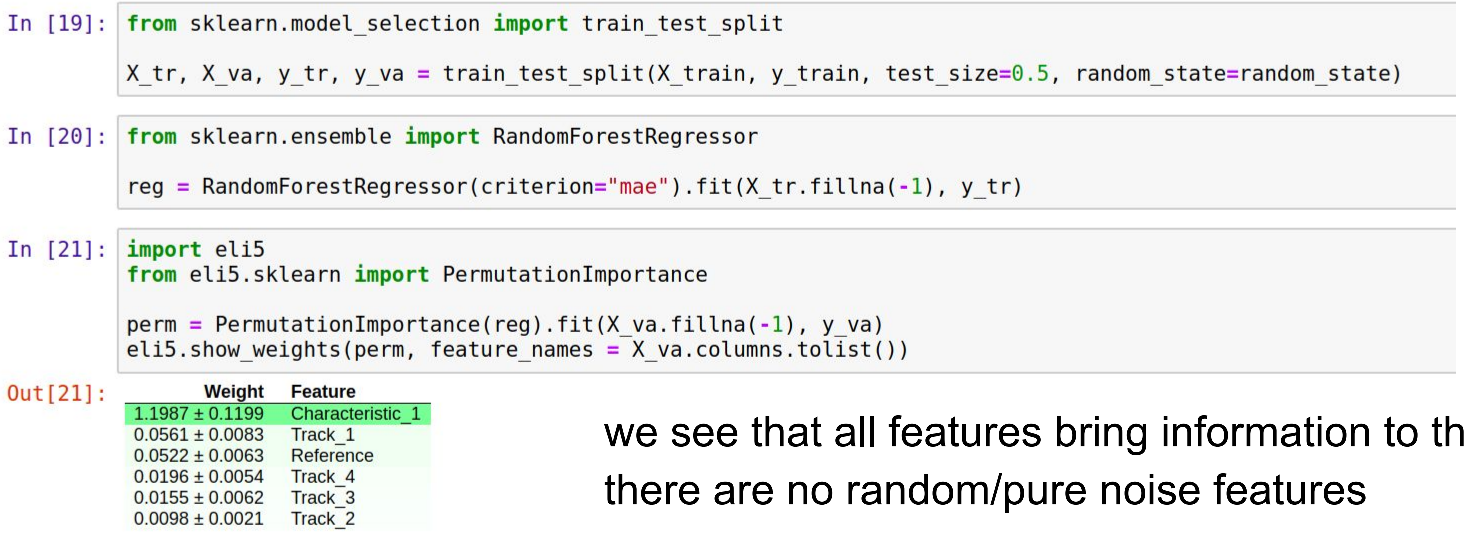
Logs



Reference	Track_1	Track_2	Track_3	Track_4	Characteristic_1
978.173	612.3953	94.66523	1.811031	15.6434	0.106912968
977.689	608.0019	93.84432	1.893699	15.73843	0.105824448
977.205	606.6861	93.17545	1.948597	15.80422	0.105495448

Permutation importance / Mean Decrease Accuracy (MDA)

black-box estimator measuring how score decreases when a feature is not available



```
In [19]: from sklearn.model_selection import train_test_split

         X_tr, X_va, y_tr, y_va = train_test_split(X_train, y_train, test_size=0.5, random_state=random_state)
```

```
In [20]: from sklearn.ensemble import RandomForestRegressor

         reg = RandomForestRegressor(criterion="mae").fit(X_tr.fillna(-1), y_tr)
```

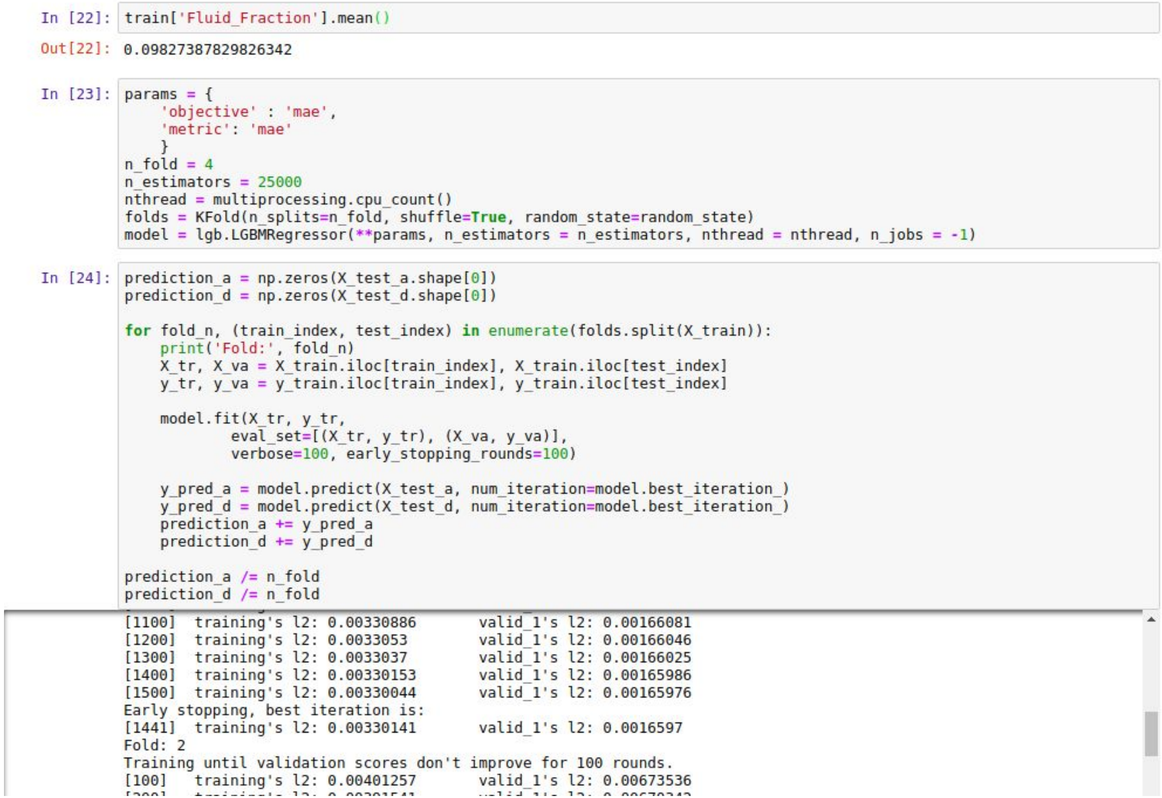
```
In [21]: import eli5
         from eli5.sklearn import PermutationImportance

         perm = PermutationImportance(reg).fit(X_va.fillna(-1), y_va)
         eli5.show_weights(perm, feature_names = X_va.columns.tolist())
```

Out[21]:

Weight	Feature
1.1987 ± 0.1199	Characteristic_1
0.0561 ± 0.0083	Track_1
0.0522 ± 0.0063	Reference
0.0196 ± 0.0054	Track_4
0.0155 ± 0.0062	Track_3
0.0098 ± 0.0021	Track_2

we see that all features bring information to the model, there are no random/pure noise features

Model



```
In [22]: train['Fluid_Fraction'].mean()

Out[22]: 0.09827387829826342

In [23]: params = {
             'objective' : 'mae',
             'metric': 'mae'
             }
         n_fold = 4
         n_estimators = 25000
         nthread = multiprocessing.cpu_count()
         folds = KFold(n_splits=n_fold, shuffle=True, random_state=random_state)
         model = lgb.LGBMRegressor(**params, n_estimators = n_estimators, nthread = nthread, n_jobs = -1)

In [24]: prediction_a = np.zeros(X_test_a.shape[0])
         prediction_d = np.zeros(X_test_d.shape[0])

         for fold_n, (train_index, test_index) in enumerate(folds.split(X_train)):
             print('Fold:', fold_n)
             X_tr, X_va = X_train.iloc[train_index], X_train.iloc[test_index]
             y_tr, y_va = y_train.iloc[train_index], y_train.iloc[test_index]

             model.fit(X_tr, y_tr,
                     eval_set=[(X_tr, y_tr), (X_va, y_va)],
                     verbose=100, early_stopping_rounds=100)

             y_pred_a = model.predict(X_test_a, num_iteration=model.best_iteration_)
             y_pred_d = model.predict(X_test_d, num_iteration=model.best_iteration_)
             prediction_a += y_pred_a
             prediction_d += y_pred_d

         prediction_a /= n_fold
         prediction_d /= n_fold
```

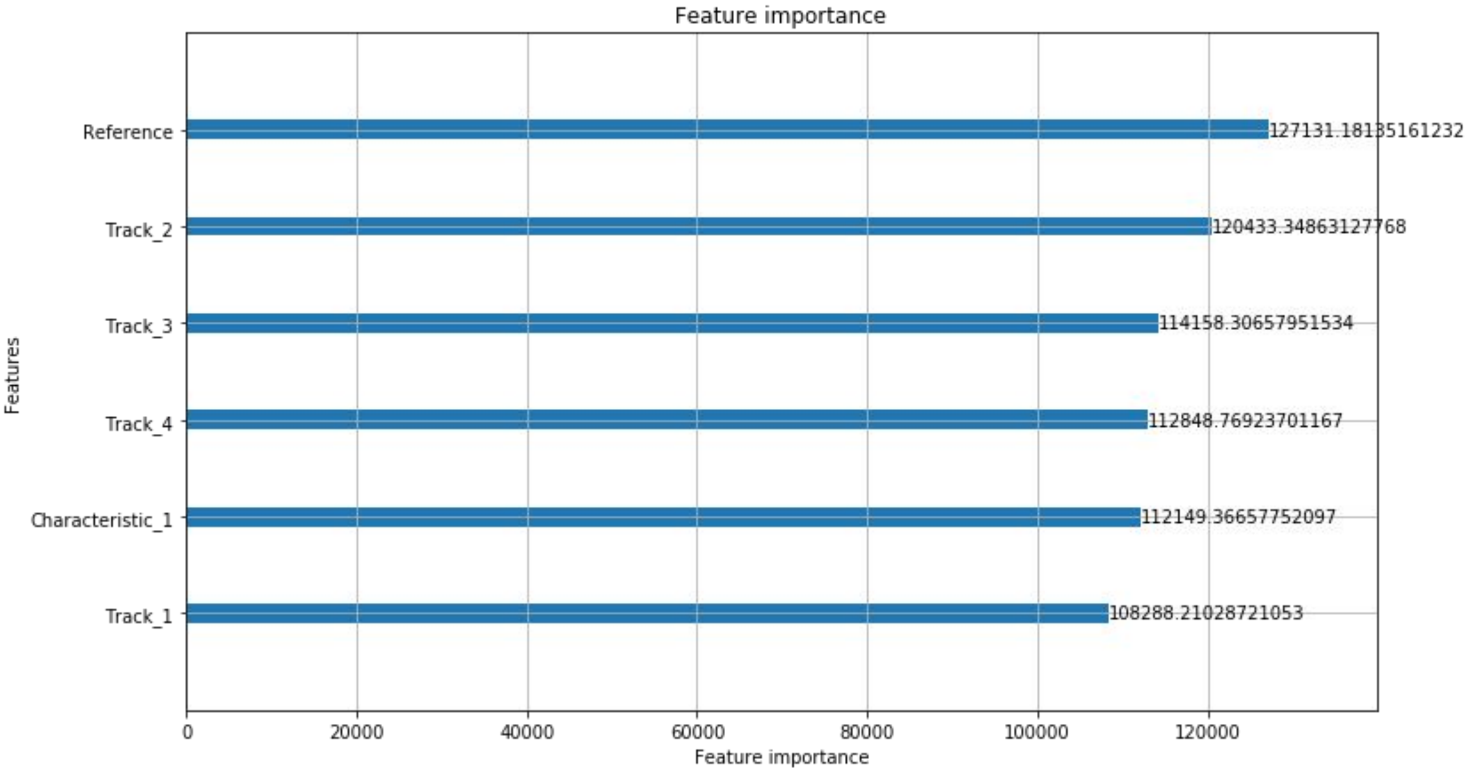
```
[1100]  training's l2: 0.00330886       valid_1's l2: 0.00166081
[1200]  training's l2: 0.0033053        valid_1's l2: 0.00166046
[1300]  training's l2: 0.0033037        valid_1's l2: 0.00166025
[1400]  training's l2: 0.00330153       valid_1's l2: 0.00165986
[1500]  training's l2: 0.00330044       valid_1's l2: 0.00165976
Early stopping, best iteration is:
[1441]  training's l2: 0.00330141       valid_1's l2: 0.0016597
Fold: 2
Training until validation scores don't improve for 100 rounds.
[100]   training's l2: 0.00401257       valid_1's l2: 0.00673536
```

Model:
Gradient boosting
Cross-validation:
4 folds
Metric:
Mean absolute error

$$mae = \frac{\sum_{i=1}^{n} abs\left(y_i - \lambda(x_i)\right)}{n}$$

early stopping to prevent overfitting

Feature importance


Feature importance
Reference
Track_2
Track_3
Track_4
Characteristic_1
Track_1
Features
Feature importance