**Capstone Project 1**

**In-depth Analysis (Machine Learning)**

The current project derived its machine learning question with an aim to imputing missing values for the depths. The hypothesis is that given some of the major variables that characterize produced waters and the basins they are present in, one can reasonably well predict the depth at which they originate. The problem is framed as a regression problem given the predicted variable (DEPTHUPPER) is continuous. In order to do that, the data needed to be prepared as follows:

1. Solve for Cl, Ca and Na using median imputation.
2. Drop null TDS values (given that there are only 12 missing).
3. Separate missing depths data out, and create a test/train dataset with the remaining variables.
4. Test and evaluate different linear regression models to solve for DEPTHUPPER using relevant variables.

The most important part of selecting features comes from background knowledge. Produced waters have characteristics determined by their origin basins, formations and absolute depths below sea level. Conversely, depths should be related to the water quality and the basins of origin. It should be noted that these waters are naturally pressurized in the reservoirs, and not the result of hydraulic fracture stimulations. As a result the major predictor variables are **latitude, longitude, basin, welltype, chlorine (Cl), calcium (Ca), sodium (Na) and total dissolved solids (TDS) concentrations**. A few more variables like magnesium and potassium-sodium concentrations could be chosen, but given the heat map generated between the continuous variables, multicollinearity development seemed apparent.

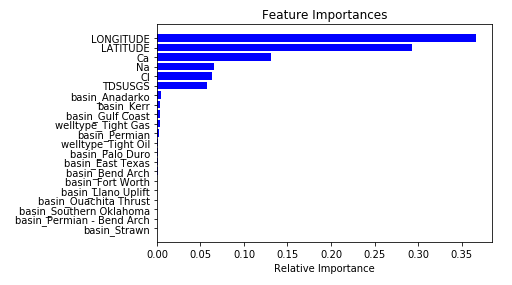
Steps 1 and 2 were easily accomplished, using *fillna()* and filtering for *notnull()* values from the TDSUSGS column. The data was filtered to include the variables mentioned above and stored in *df\_prep*. For the third step, the missing depths were easily filtered out (4946 values out of total 16619 values). Using the *sklearn.metrics.train\_test\_split()* function, 20% of the data was separated then as test data. The rest of the training data was fitted with different regression models.

One of the challenges of handling the data was the presence of categorical types like basin (15 types) and well type (3 kinds). Formations were initially meant to be used as a predictor variable, but there are 2029 unique formations available, and so the idea was not pursued. To handle the categorical data, both variables were converted to dummy variables, with 1 less variable each (to avoid multicollinearity and redundancy). Then they were joined with the train data. The other variables were then standardized with a mean of 0 and a standard deviation of 1 using the *sklearn.preprocessing.StandardScaler()* function.

Regression evaluation takes the form of accuracy scores (R2 values between the predicted and real target variables), or root mean squared errors (RMSE). The results for the various regressions are denoted as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Parameters** | **Training R2 value** | **Test R2 value** | **RMSE** |
| Ordinary Least Squares | *Default* | 0.098 | 0.098 | 2789.5 |
| Lasso Regression | *alpha = 0.001, max\_iter = 10000* | 0.098 | 0.098 | 2789.5 |
| Ridge Regression | *alpha = 0.001, max\_iter = 10000* | 0.098 | 0.098 | 2789.5 |
| Stochastic Gradient Descent Regression | *max\_iter=1000, tol=1e-4, eta0=0.1* | 0.091 | 0.089 | 2803.0 |
| Linear Support Vector Machines Regression | *epsilon=.01* | -0.236 | -0.237 | 3266.5 |
| Polynomial Support Vector Machines Regression | *kernel='poly', degree=2, C=100, epsilon=1* | 0.060 | 0.064 | 2842.2 |
| Decision Tree Regression | *max\_depth=15* | 0.861 | -0.139 | 3134.8 |
| Random Forest Regression | *max\_depth=20, n\_estimators=100* | 0.947 | 0.428 | 2221.3 |
| Gradient Boosting Regression | *max\_depth=20, n\_estimators=50, learning\_rate=1* | 1.000 | -0.213 | 3235.7 |
| k-Nearest Neighbors Regression | *n\_neighbors=5* | 0.680 | 0.468 | 2142.2 |

The results were not initially the greatest, and a feature selection was performed using the random forest regressor’s in-built *feature\_importances\_* attribute. Only about 6 features were revealed to have any major effects on the target variable, DEPTHUPPER. This finding was interesting and running the regression with only the top 10 features gave similar results, i.e. 0.948 for train score data, and 0.426 for test score data and 2224.8 for RMSE.



Standardization of the latitudes and longitudes doesn’t make much sense, and so the unscaled datasets were run through the top regressors of scaled data, i.e. Random Forest, k-NN and gradient boosting regressors. The results were much better, and are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Parameters** | **Training R2 value** | **Test R2 value** | **RMSE** |
| Random Forest Regression | *max\_depth=25, n\_estimators=100* | 0.958 | 0.745 | 1483.8 |
| Gradient Boosting Regression | *max\_depth=20, n\_estimators=50, learning\_rate=1* | 1.000 | 0.497 | 2083.9 |
| k-Nearest Neighbors Regression | *n\_neighbors=5* | 0.410 | 0.122 | 2752.7 |

Running a 5-fold cross-validation on the best regressor (Random Forest), a *max\_depth* of 30 and *n\_estimators* of 150 were discovered to be optimal, giving 0.960 for train score data, and 0.745 for test score data and 1478.4 for RMSE.

Based on this data, a prediction was performed for the missing depths and the missing values replaced. An interactive Bokeh plot was then constructed as follows, outlining the depths versus TDS relationships.

