**Background:**

Well logs are interpreted/processed to estimate the in-situ petrophysical and geomechanical properties, which is essential for subsurface characterization. Various types of logs exist, and each provides distinct information about subsurface properties. Certain well logs, like gamma ray (GR), resistivity, density, and neutron logs, are considered as “easy-to-acquire” conventional well logs that are run in most of the wells. Other well logs, like nuclear magnetic resonance, dielectric dispersion, elemental spectroscopy, and sometimes sonic logs, are only run in limited number of wells.

Sonic travel-time logs contain critical geomechanical information for subsurface characterization around the wellbore. Often, sonic logs are required to complete the well-seismic tie workflow or geomechanical properties prediction. When sonic logs are absent in a well or an interval, a common practice is to synthesize them based on its neighboring wells that have sonic logs. This is referred to as sonic log synthesis or pseudo sonic log generation. [1]

**Summary:**

Data with varying log properties is provided from the VOLVE dataset owned by Equinor. The data is from two neighboring wells, referred to as Well 1 and Well 2. Table 1 shows the log data provided from the VOLVE dataset. A yes denotes that the well was provided with that style of log for analysis.

Table 1 - Log Availability for Provided Wells

|  |  |  |  |
| --- | --- | --- | --- |
| Log Name | Units | Well 1 | Well 2 |
| Caliper (CAL) | Inches | Yes | Yes |
| Neutron (CNC) | Dec | Yes | Yes |
| Gamma Ray (GR) | API | Yes | Yes |
| Deep Resistivity (HRD) | Ohm•m | Yes | Yes |
| Medium Resistivity (HRM) | Ohm•m | Yes | Yes |
| Photoelectric Factor (PE) | Barn | Yes | Yes |
| Density (ZDEN) | Gram/m3 | Yes | Yes |
| Compress Travel time (DTC) | ns/foot | Yes |  |
| Shear Travel time (DTS) | ns/foot | Yes |  |

The goal of this study is to select and train a machine learning model from Well 1 to predict compressional travel time and shear travel time. Once the model is trained it is deployed onto well 2 to predict the same features.

The study follows the format of a typical machine learning assessment. Data is imported, cleaned, inspected, features are ranked, algorithms are spot checked, data is preprocessed, hyperparameters are tuned. The machine learning model is optimized, trained across the entire data set from well 1 and deployed onto well 2. Table 2 provides a summary of the models selected.

Table 2 - Models selected for testing

|  |  |  |
| --- | --- | --- |
| Name | Abbreviation | Description |
| Linear Regression | LinReg | Baseline |
| Ridge Regression | Ridge | Regularization |
| Lasso Regression | Lasso | Regularization |
| K Nearest Neighbors | KNN | Baseline |
| Decision Tree | DecTree | Baseline |
| Support Vector | SVR | Baseline |
| Multi-Layer Perceptron | MLP | Neural Net |
| Ada Boost Regressor | AB | Ensemble |
| Gradient Boosting | GBM | Ensemble |
| Random Forest | RF | Ensemble |
| Extra Trees Regressor | ET | Ensemble |

It is important to note a wide range of models were selected. This was to test the model’s ability to handle variations in data, recognize patterns and optimize bias/variance. For neural networks, a simplistic multi-layer perceptron is selected. The MLP was outperformed by KNN, random forest and extra trees. The classification and regression trees (CART) models are less computationally expensive, more transparent and the hyperparameters can easily be tuned. No further investigation into a neural network was completed.

The three best standardized data models, KNN, random forest and extra trees, were selected for hyper parameter tuning. Post hyper parameter tuning, the extra tree model had the best performance. After training the model on Well 1’s data, it was deployed to make predictions on Well 2’s features.

**Data Inspection:**

Data features are graphed against target variable compressional travel time (DTC). The analysis noted that the two target variables, compressional travel time (DTC) and shear travel time (DTS) had a strong Pearson correlation coefficient of 0.95. A heatmap of the Pearson correlation coefficients for the data is displayed in figure 1. For initial inspection it was deemed unnecessary to compare feature variables to both target variables, as they did not appear to be independent. Finalized model creation did include both target variables.

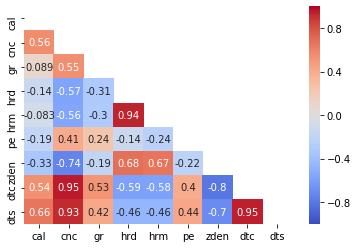


Fig. 1 Heatmap of Pearson Correlation Coefficient

From initial inspection a few statistical outliers were noted, which moderately threw off model accuracy. The data is placed through a z-score filter to remove all variables that are greater than three standard deviations from the mean. This maintains data shape while focusing the data in core areas. The photo electric factor (PE) had an unusual amount of data that was below 0.75, referred to as an anomaly. A joint probability mass function (PMF), probability density function (PDF) and scatter plot vs compressional travel time (DTC) is graphed in figure 2.

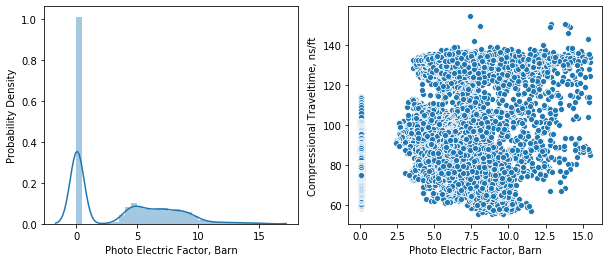


Fig. 2 Mass, Density and Scatter Plot of Photo Electric Factor

To understand the effect of the photo electric factor anomaly, three different datasets were created. First set; all the data was maintained expect for gross outliers, second set; observations with PE factor greater than 0.75, and third set; observations with PE factor less than 0.75. Spot checking algorithms with the different datasets showed little variation in model quality. Dataset 1 which contained all the data was kept for simplicity. With the dataset defined a base algorithm needed to be performed to better compare model performance.

**Algorithm Performance, No pre-processing**

Seven models are selected for early testing. They are linear regression, ridge regression, lasso regression, k-nearest neighbors, decision tree, support vector regression and multilayer perceptron. Ensemble methods are not selected in the process. Decision trees give an indication of random forest and extra trees performance, while regression gives an indication of boosting performance. The base performance results are shown in figure 3. The results are obtained by running each model on the same dataset with k-folds cross validation equal to ten. Box and whisker plots are created to show the results of the ten runs.

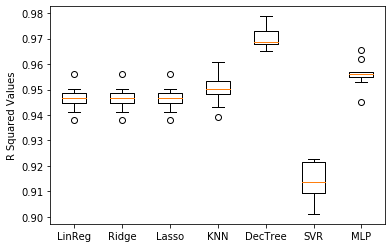


Fig. 3 Machine Learning Algorithm Base Performance

Decision tree is the initial front runner, followed by multi-layer perceptron, and k-nearest neighbors. The regression and regularization models did not perform as well. Lasso and ridge regression were created with an alpha parameter of 0.002, supporting why the performance was so close to that of the baseline regression model.

**Algorithm Performance, Standardized Data**

Models such as k-nearest neighbors and support vector regression are known to have significant model improvement when working on a standardized dataset. This involves transforming all the feature vectors to have their mean centrally located at zero and a standard deviation of 1. The same method of k-folds cross validation with 10 splits is used. The results of model performance on standardized datasets is shown in figure 4.

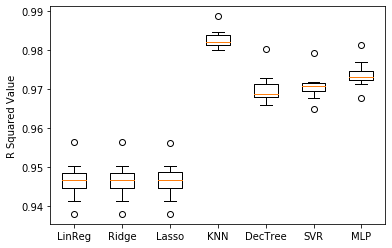


Fig. 4 Machine Learning Algorithm Performance Standardized Datasets

As seen, k nearest neighbors, support vector regression and multilayer perceptron all experienced improvements on model performance. Support vector regression (SVR) had the largest increase in performance from 0.914 to 0.971. KNN was selected for further testing with a r-squared value of 0.983.

**Hyperparameter Tuning, KNN**

The default k nearest neighbor model from scikit learn (sklearn) uses voting logic with the five nearest neighbors and the neighbor weights are uniform. Three different algorithms are available for selection, which are ball tree, kd tree, and brute [2]. Figure 5 shows how the r squared value changes with varying neighbor values from one to twenty-one. The increments are kept as odd numbers to prevent ties from occurring during the voting process.

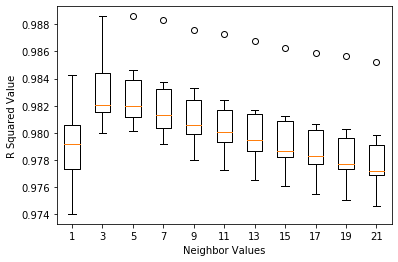


Fig. 5 KNN performance with neighbor value tuning

From figure 5 it is evident that neighbor value of three will provide the best fit. A unified graph was not created to assess the difference between the KNN algorithm and voting weights. The sklearn model selection function *GridSearchCV* [3]was used to select the best parameters, which showed ball tree for the algorithm and distance for the weights. From optimizing the KNN hyperparameters, algorithm = ball tree, n\_neighbors = 3, weights = distance, the r squared value was increased to 0.984.

**Ensemble Model Evaluation**

Four ensemble models, Ada Boost Regression, Gradient Boosting Regression, Random Forest Regression and Extra Tree Regression, are selected for model building. Random Forest and Extra Tree are selected based on the proficient performance of the decision tree model from previous steps. The boosting methods are selected for the authors personal experience and evaluation. Figure 6 shows how the different models performed with no hyperparameter tuning of any. They were produced using 10 splits from k folds cross validation. Each feature was standardized to have a mean of zero and standard deviation one.

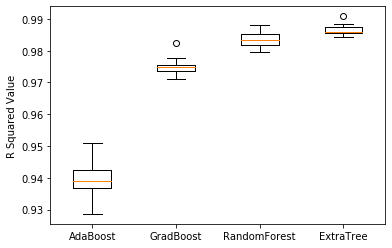


Fig. 6 Ensemble Model Performance Standardized Datasets

From figure 6 the extra tree model provides the best base performance of the ensemble models with a r-squared value of 0.986. The model was selected for hyperparameter tuning.

**Hyperparameter Tuning, Extra Trees**

The hyper parameter with the greatest effect on ensemble tree-based models is the number of trees generated. As the number of trees is increased the model performance increases as well. This also requires more computational power to account for the additional tree models. The number of trees was varied from five to 300. The computation time was tracked to see if a tradeoff of accuracy for speed was warranted. The results from the hyperparameter tuning are shown in figure 7.

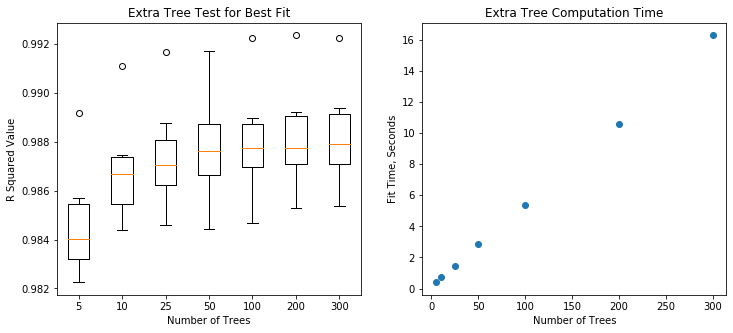


Fig. 7 Hyperparameter Turning Extra Trees Model

The model performance hits an asymptote with 200 to 300 trees, while the computation time maintains a linear upward trend. 300 trees are selected as they provide a good balance of model accuracy and efficiency to build the model across. With 300 trees an average r squared value of 0.988 is obtained. The extra trees model is marginally better than the KNN model with a r squared value of 0.983.

With a good base of model evaluations and hyper parameter tuning. The extra trees model can be trained across the entire dataset from well one and have a well log synthesized for well two.

**Finalized Training and Deployment**

The extra trees model was trained across all the observations from the standardized dataset. A k folds cross validation was performed with 10 splits, resulting in an r squared value of 0.9905 and a root mean square error of 4.7182**.** To visualizethe model performance a single 80/20 train test split is produced. Figure 8 shows the resulting predictions of the 80/20 split for DTC and DTS.

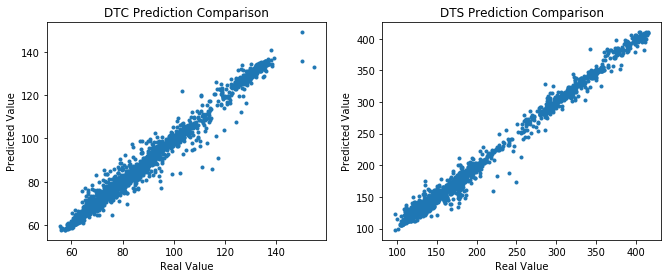


Fig. 8 Graphical Results of Single 80/20 Prediction with Extra Trees

From figure 8 it is apparent that the extra trees model does a good job of predicting the values for DTC and DTS from the input feature columns. The data from well two is imported and predictions are generated. These predictions can be seen in figure 9. Since the actual values are not released, then no methodology exists for understanding goodness of fit.

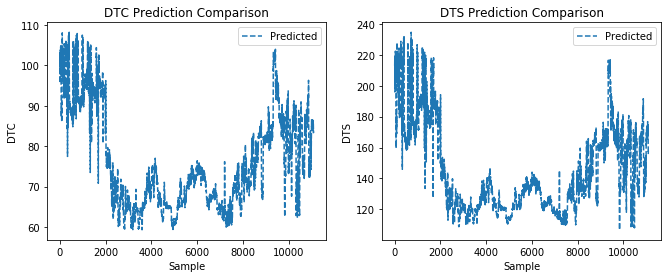
****

Fig. 9 Predicted values for DTC and DTS from Well 2

**Summary**

A total of eleven machine learning models were investigated on their ability to predict compressional travel time and shear travel time. With a standardized dataset KNN performed the best of the baseline models. Upon introducing ensemble models’ extra trees performed the best. Both KNN and Extra Trees had their hyperparameters tuned, resulting in R squared values of 0.983 and 0.988, respectively. The Extra trees model was ultimately selected, trained across the entire well 1 dataset and deployed to predict values for well 2. The actual values from well 2 will be released later.

**References**

[1] Yu, Yanxiang, 2020, *Synthetic Sonic Log Generation*, SPWLA, viewed 20 April 2020, <<https://github.com/pddasig/Machine-Learning-Competition-2020/blob/master/Synthetic%20Sonic%20Log%20Generation%20Starter_Yu%202_27_2020.ipynb>>

[2] Scikit Learn, 2019, *K Nearest Neighbors Documentation,* sklearn, viewed 20 April 2020*, <*<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html#sklearn.neighbors.KNeighborsRegressor>>

[3] Scikit Learn, 2019, *GridSearchCV Documentation*, sklearn, viewed 20 April 2020, <<https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>>