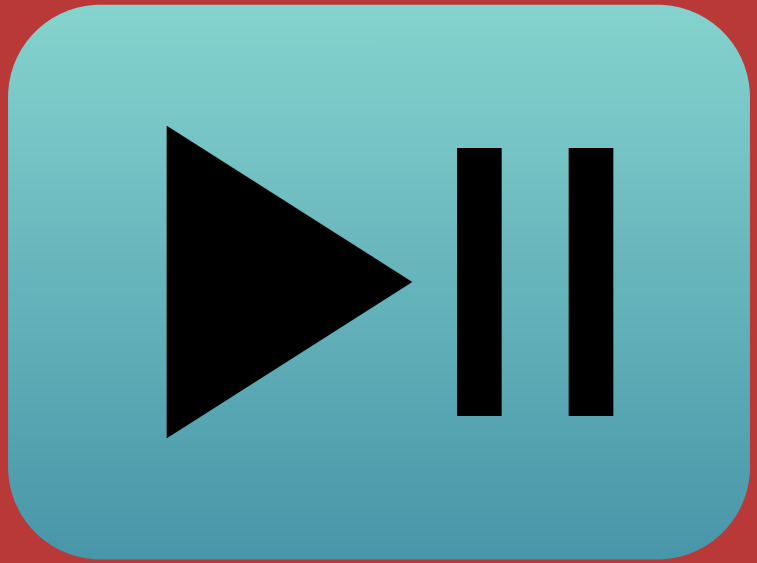
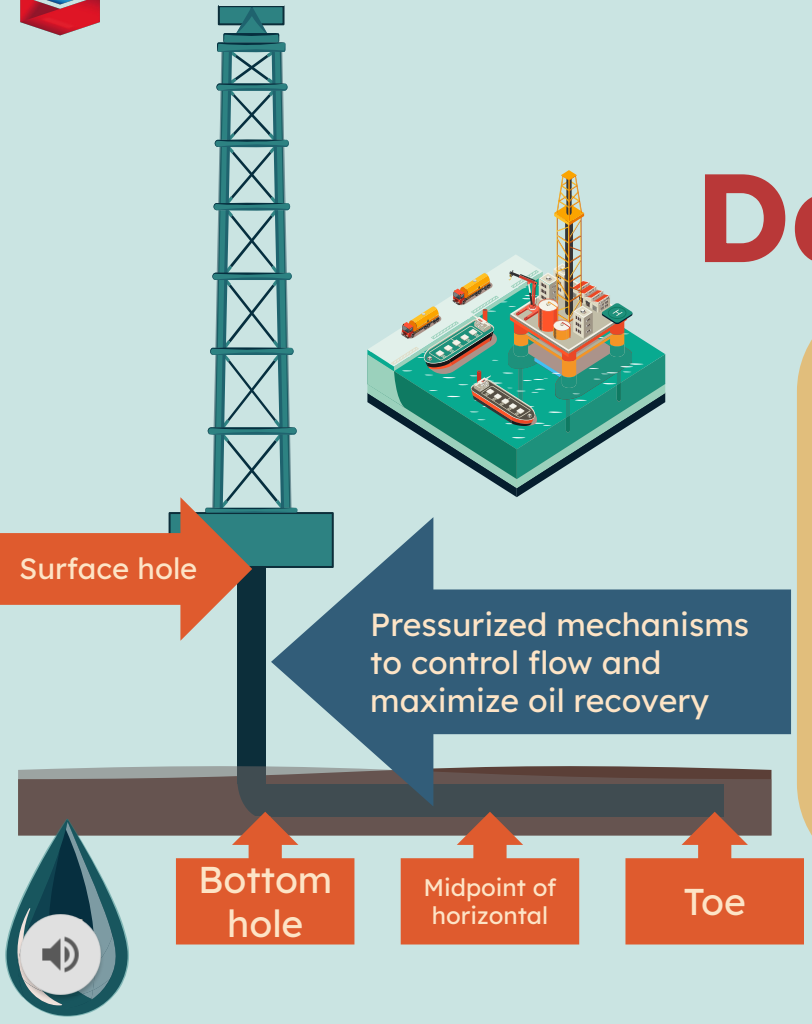
Rice Datathon 2024

Rohan Chaudhary, Zahra Bukhari, Caleb C. Hairston, and Kishan Yerneni



We encourage you to pause the presentation and read as needed

Background & Data Significance



A major part of data science is understanding the data and variables we have, so we can use them as a basis to reach conclusions. It is easy to jump into the technical aspect of data science and overlook this key knowledge, so our team made it a point to seek out these crucial details.

Coming into this Datathon, our team had no knowledge of how oil collection worked. We spoke with the helpful Chevron mentors to clarify the variables we were given. This insight was integral to our processing and conclusions.

Initial Approach

1. Visualized given data
2. One-hot encoding for feature engineering
3. Created baseline models to predict oil peak rate given asset data
    - Linear Regression
    - Random Forest
    - XGBoost
    - Lasso
4. Clean and create resultant models

Initial Approach: Results

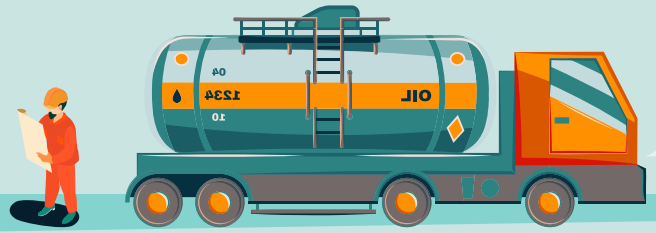
Baseline models

- Best Initial Root Mean Squared Error (RMSE): ~96.3 using Lasso Regression Model.
- Regardless of the RMSE value, the training sample size ~1300, which is significantly lower compared to the initial size of ~29000.
- Disparity in sample sizes could lead to overfitting issues in the models.

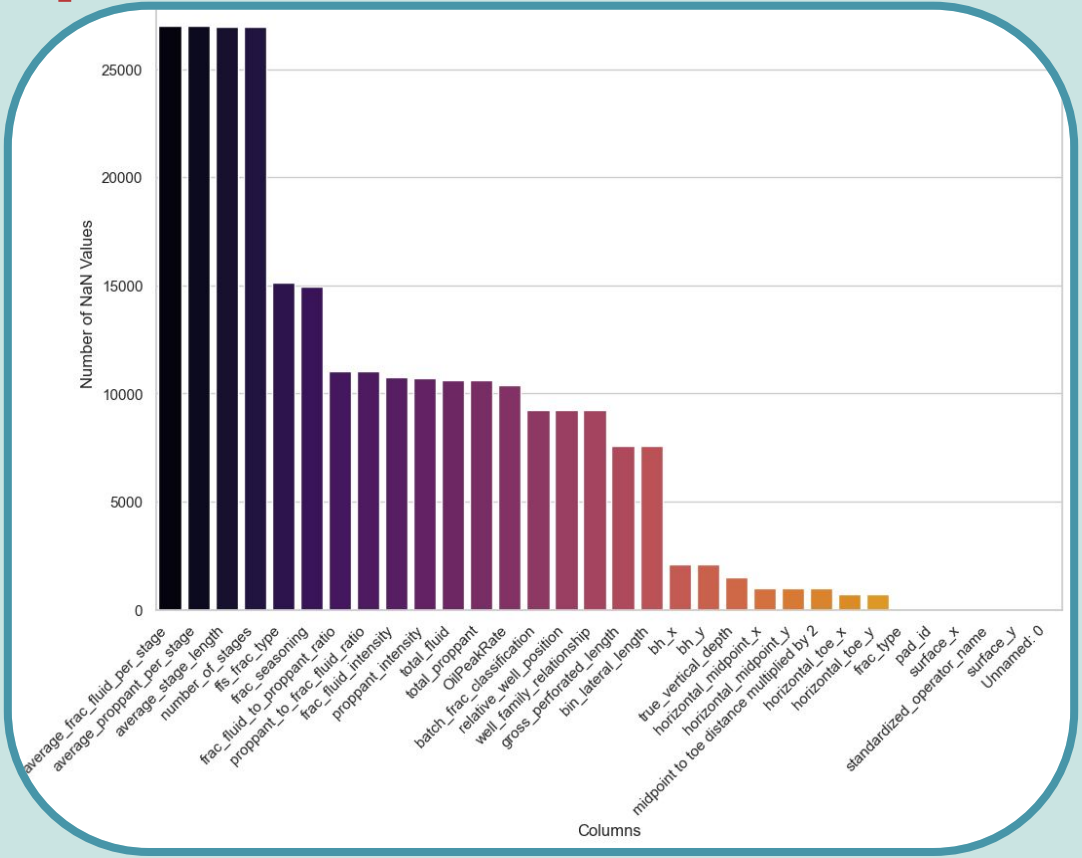
```
from sklearn.metrics import mean_squared_error, r2_score
selected_features = np.array(X.columns)[coef != 0]

# Refit Lasso model with selected feature, using best alpha value from grid search
lasso_model_selected_features = Lasso(alpha=1.3)
lasso_model_selected_features.fit(X_train[selected_features], y_train)

# Evaluate the model on test data
predictions = lasso_model_selected_features.predict(X_test[selected_features])
mse = mean_squared_error(y_test, predictions)
r_squared = r2_score(y_test, predictions)

print("MSE:", mse)
print("R-squared:", r_squared)
```

```
MSE: 9284.622710025973
R-squared: 0.45818523500945185
```

Lasso Regression Code Snippet

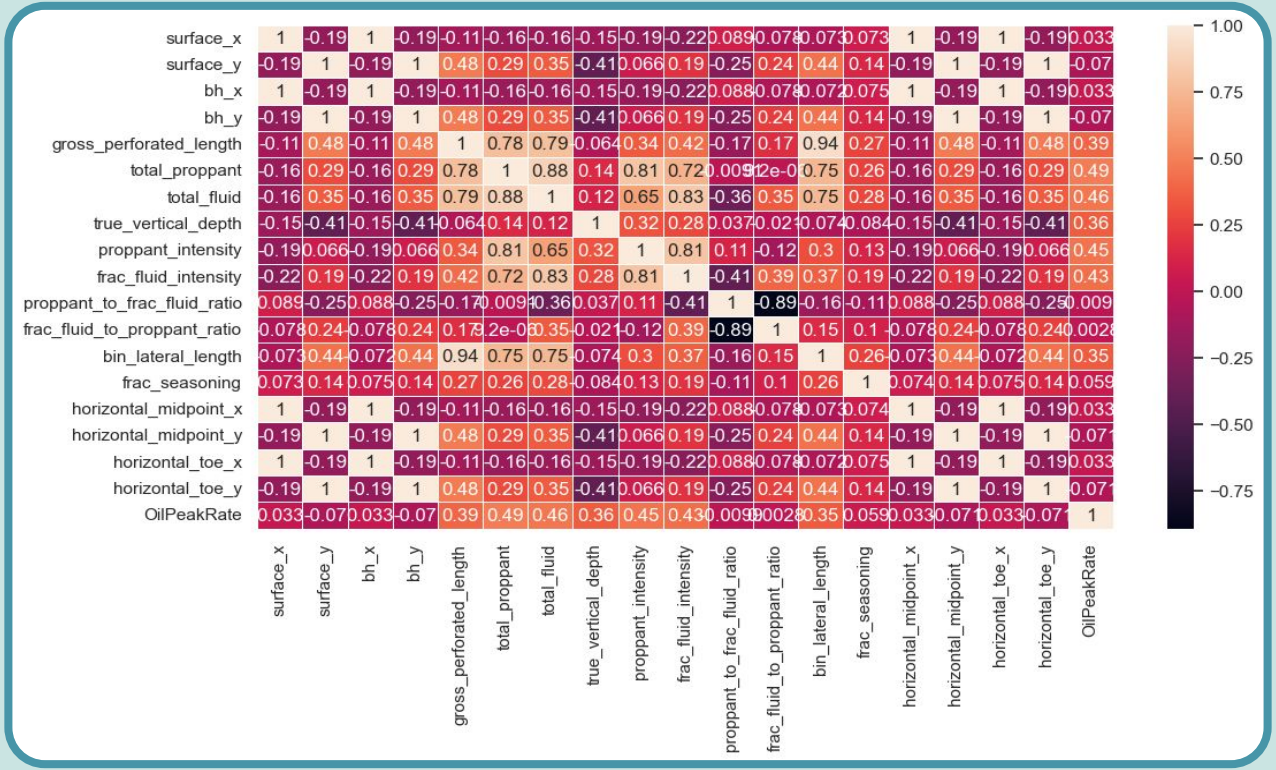


Exploration & Visualization



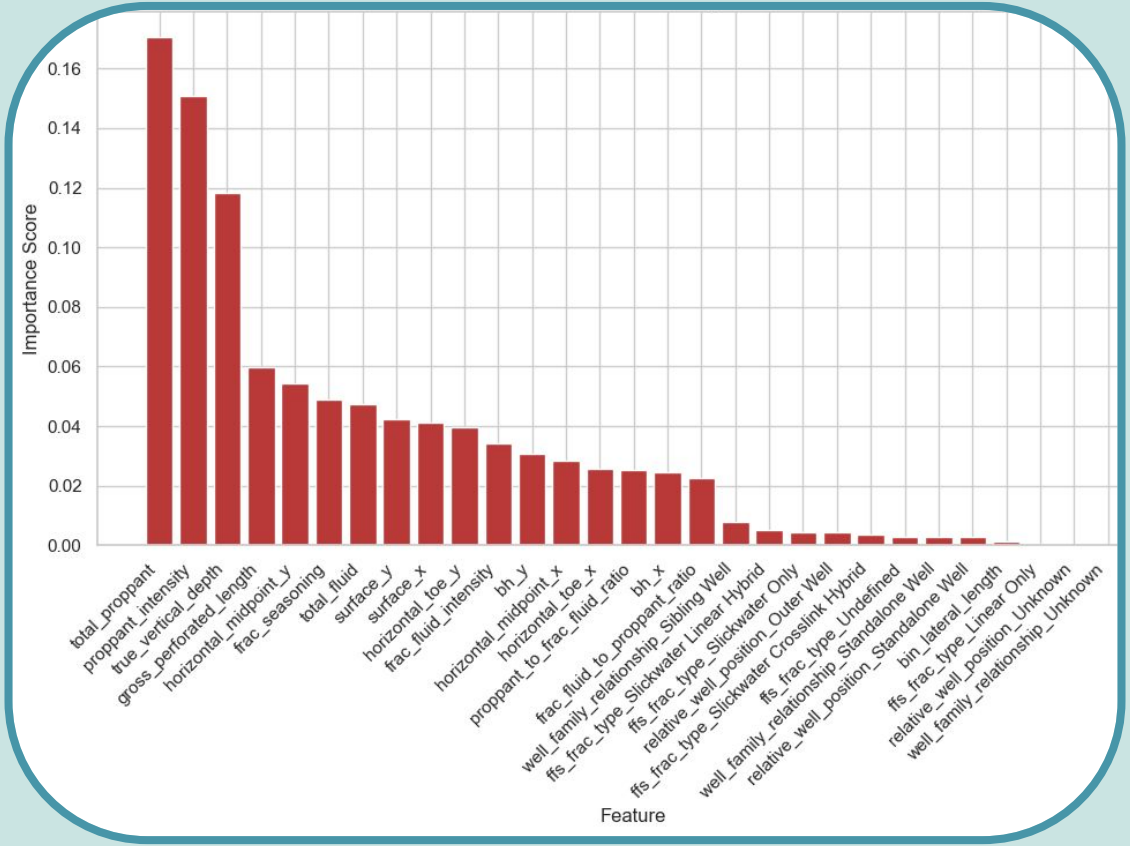
Number of NaN Values per Column

Exploration & Visualization



Heat Map

Exploration & Visualization



Random Forest Feature Importance

Reevaluation and Final Approach

1. Analyzed findings from Number of NaN Values per Column graph and made the decision to drop columns with NaN values of over 70% of total data, instead of rows, leading to a larger sample size after cleaning data.
2. Decided which columns were insignificant enough to remove using Random Forest Feature Importance.
3. Dropped outliers beyond three standard deviations from the mean.
4. Imputed Data: Used linear regression and KNN to fill in remaining null column values.

Results and Conclusion

By utilizing these combined data science techniques:

- Dropping outliers and columns with null value sum > 70%
- Using graphs to evaluate and drop insignificant columns
- Imputing data to fill in the rest of the NaN values

We achieved a RMSE of **~31** with the cleaned initial given data, using the Random Forest Regression model.

Although we obtained lower RMSEs from dropping large amounts of null data, we settled for this method as our final RMSE. We concluded the models that resulted from dropping large amounts of data, to produce a smaller RMSE, were not reflective of the original dataset. As mentioned earlier, this practice could risk overfitting the model.

*LOWER RMSE ACHIEVED SINCE RECORDING

```
17
18  rf = RandomForestRegressor(n_estimators=100, random_state=45)
19
20  X_train, X_test, y_train, y_test = train_test_split(df_1.drop('OilPeakRate', axis=1), df_1['OilPeakRate'
21                                                      test_size=0.2, random_state=101)
22
23  rf.fit(df_1.drop('OilPeakRate', axis=1), df_1['OilPeakRate'])
24
25  joblib.dump(rf, 'rf_scaled.pkl')
26
27  rf_pred = rf.predict(X_test)
28
29  rf_pred_train = rf.predict(X_train)
30
31  mse = mean_squared_error(y_test, rf_pred)
32  mse_train = mean_squared_error(y_train, rf_pred_train)
33
34  rmse = np.sqrt(mse)
35  train_rmse = np.sqrt(mse_train)
36
37  print('RMSE of Training set:', train_rmse)
38  print('RMSE of Testing set:', rmse)
```

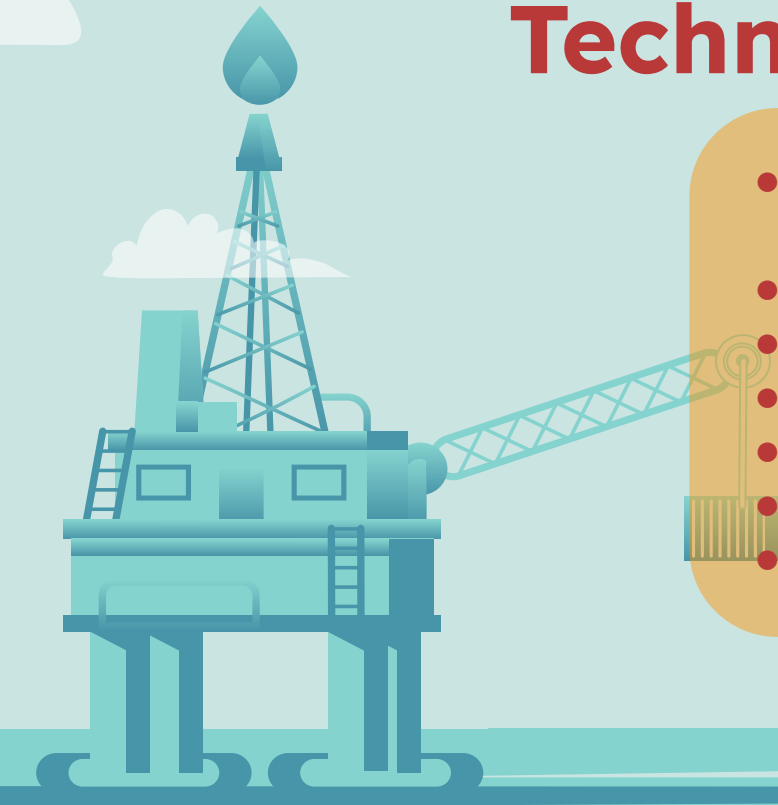
✓ 1m 48.5s

```
RMSE of Training set: 32.44589485648686
RMSE of Testing set: 31.373576156230197
```

Random Forest Regression Code Snippet

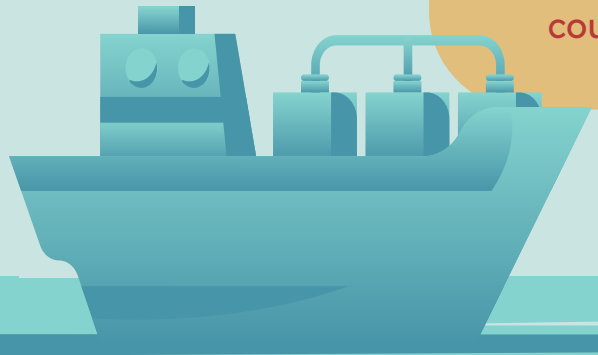
Technologies Used

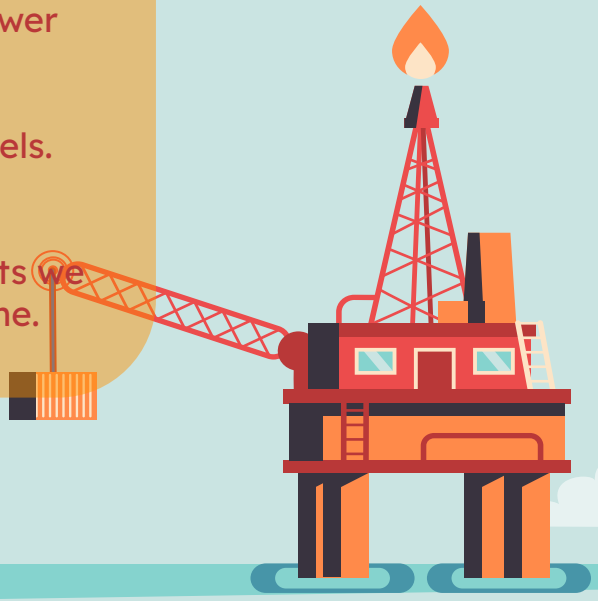
- Jupyter Notebook (Python)
- Sci-kit Learn
- Tensorflow
- XGBoost
- Numpy/Pandas
- VS Code
- Github



Potential Improvements

- In the real world, the large amounts of null data would have been caused by locations failing to send full rows of data. If we were able to ask for data reentry to fill NaN values, this could lower our final RMSE.
- Additionally, we could have tried more models.
- Finally, the limitations of our technology definitely had an impact on the feasible tests we could run within the given 37 hour time frame.





Thanks!

Thank you to Rice Datathon organizers, sponsors, mentors, and competitors for this amazing opportunity!

CREDITS: Outside assets in this presentation were taken from Slidesgo, Flaticon, and Freepik