Task 2: Linear and Multivariable Regression

Members: Lars Olav Thorbjørnsen, Stein Are Årsnes og Sanjai Vijayaratnam

Abstract

Linear and multivariable regression models were developed in this task to estimate Vp based on log data. Linear regression models tested each feature's prediction for Vp independently, while multivariable regression uses multiple features at a time to increase accuracy. With the use of model performance metrics such as mean squared error (MSE), root mean square error (RMSE) and R^2 we evaluate the predictive quality of each approach. Results indicated that multivariable regression provided a more precise Vp estimation, with (Vs, DEN) and (Vs, DEN, NEU) being the most accurate.

Introduction

Regression analysis is a key technique in ML, often used to model relationships between variables. Regression is used to analyze the relationship between a target variable and one or more predictor variables. "The objective is to determine the most suitable function that characterizes the connection between these variables" [1]. This task applies linear and multivariable regression models to estimate Vp from the selected log files. This task was essential for understanding how different regression approaches perform in predicting Vp.

In [30]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_excel('CleanedFeatureSelectedFiltered.xlsx')
```

Importing all necessary libraries and loading the data from an Excel file to a pandas DataFrame.

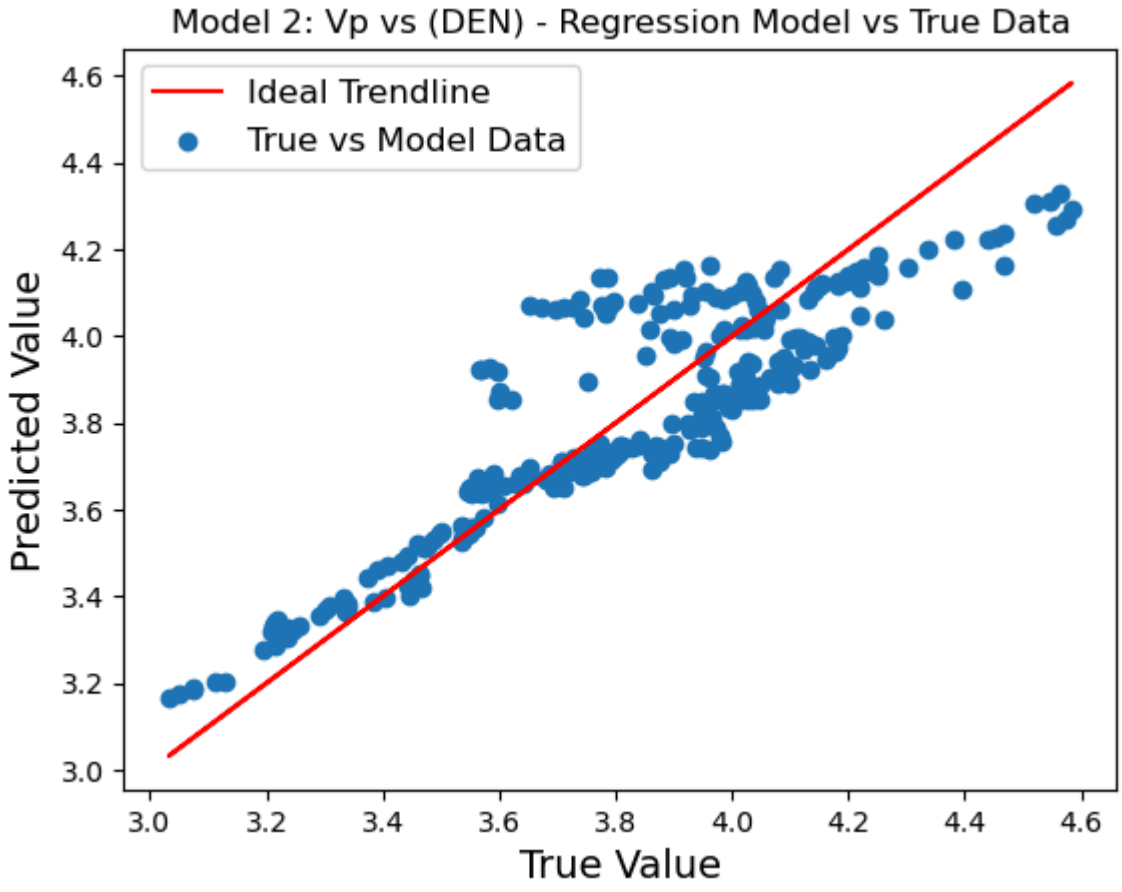
Linear Regression

In [31]:
```
def Reg(X, y, text):
    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.3, rand
    regression = LinearRegression()
    regression.fit(x_train, y_train)
    y_pred = regression.predict(x_test)
    print(regression.coef_, regression.intercept_)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)
```

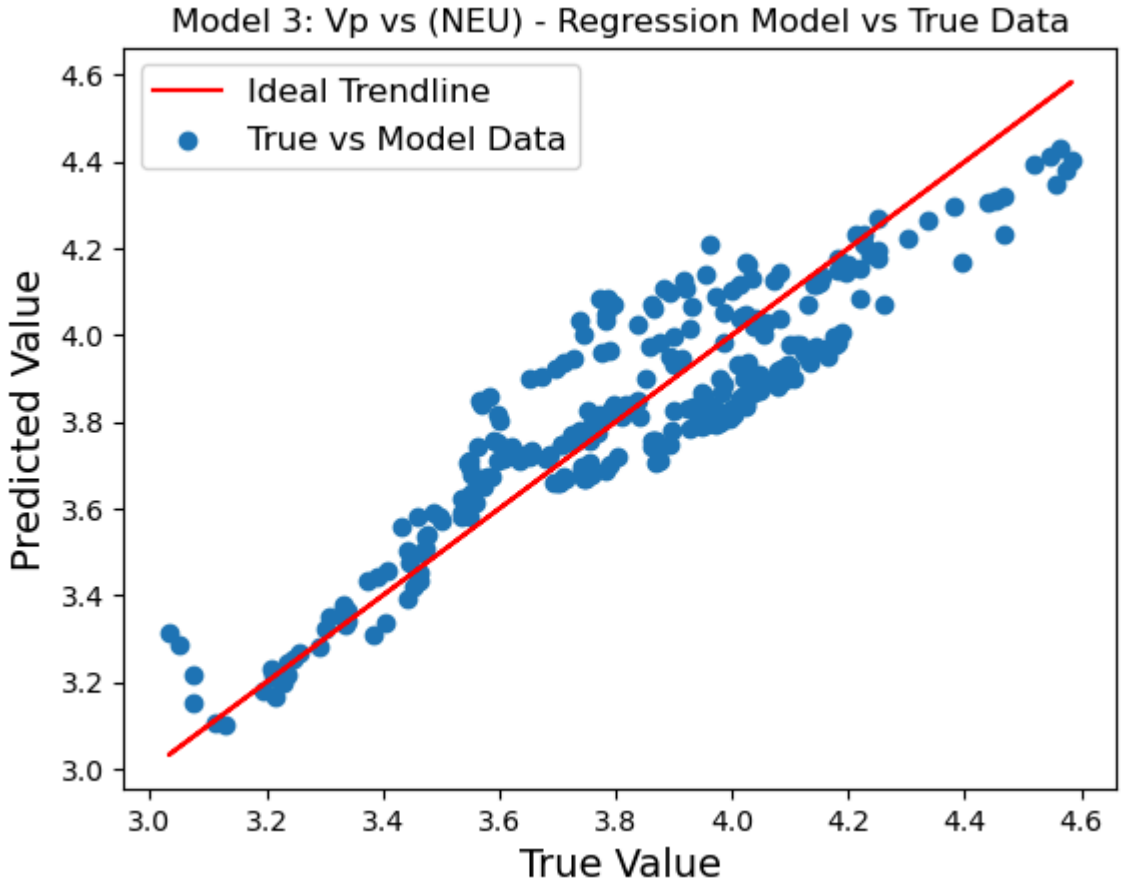
```
    print(f'{text}')
    print('Mean Squared Error:', mse)
    print('Root Mean Squared Error:',rmse)
    print('R-Square:',r2 )

    plt.plot(y_test,y_test, '-r')
    plt.scatter(y_test, y_pred)

    n = len(y_pred)
    p = X.shape[1]
    adjusted_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
    print("Adjusted R-Square:", adjusted_r2)

    plt.title(f'{text} - Regression Model vs True Data')
    plt.xlabel('True Value',fontsize=14)
    plt.ylabel('Predicted Value',fontsize=14)
    plt.legend(['Ideal Trendline', 'True vs Model Data'], loc='upper left', font
    plt.show()

y = df['Vp']
X = df[['Vs']]
Reg(X, y, "Model 1: Vp vs (Vs)")

X = df[['DEN']]
Reg(X, y, "Model 2: Vp vs (DEN)")

X = df[['NEU']]
Reg(X, y, "Model 3: Vp vs (NEU)")
```

```
[1.70152877] 0.34695867103961797
Model 1: Vp vs (Vs)
Mean Squared Error: 0.03047353102421406
Root Mean Squared Error: 0.17456669506012326
R-Square: 0.6877067952761378
Adjusted R-Square: 0.6866692763235669
```


```
[2.70846591] -2.6929728100190142
Model 2: Vp vs (DEN)
Mean Squared Error: 0.02138542281443235
Root Mean Squared Error: 0.14623755610113412
R-Square: 0.7808418650340465
Adjusted R-Square: 0.7801137649178805
```



```
[-5.2164696] 4.944747192340967
Model 3: Vp vs (NEU)
Mean Squared Error: 0.016630516874624374
Root Mean Squared Error: 0.1289593613299336
R-Square: 0.8295702126916662
Adjusted R-Square: 0.8290040007736983
```



We have made a function for linear regression that starts off with splitting the data into a training part and a testing part, with 70% for training and 30% for testing. Setting random_state=42 ensures the split is consistent each time the code runs. We then start training a linear regression model on the set (x_train, y_train). The model then predicts y_pred values for the test set. We print the coefficients (regression.coef_) and intercept (regression.intercept_) of the model, representing the linear relationship found. We then have some error measurements to evaluate the model, mean squared error (MSE) and root mean squared error (RMSE). These values measures the average squared differences between predicted and actual values. We also measure R-squared which indicates how well the data fit in the regression model [2]. Adjusted R-squared accounts for the number of predictors, making it a more reliable metric for models with multiple predictors. In the end we make a plot to compare the predicted values with the true test values.

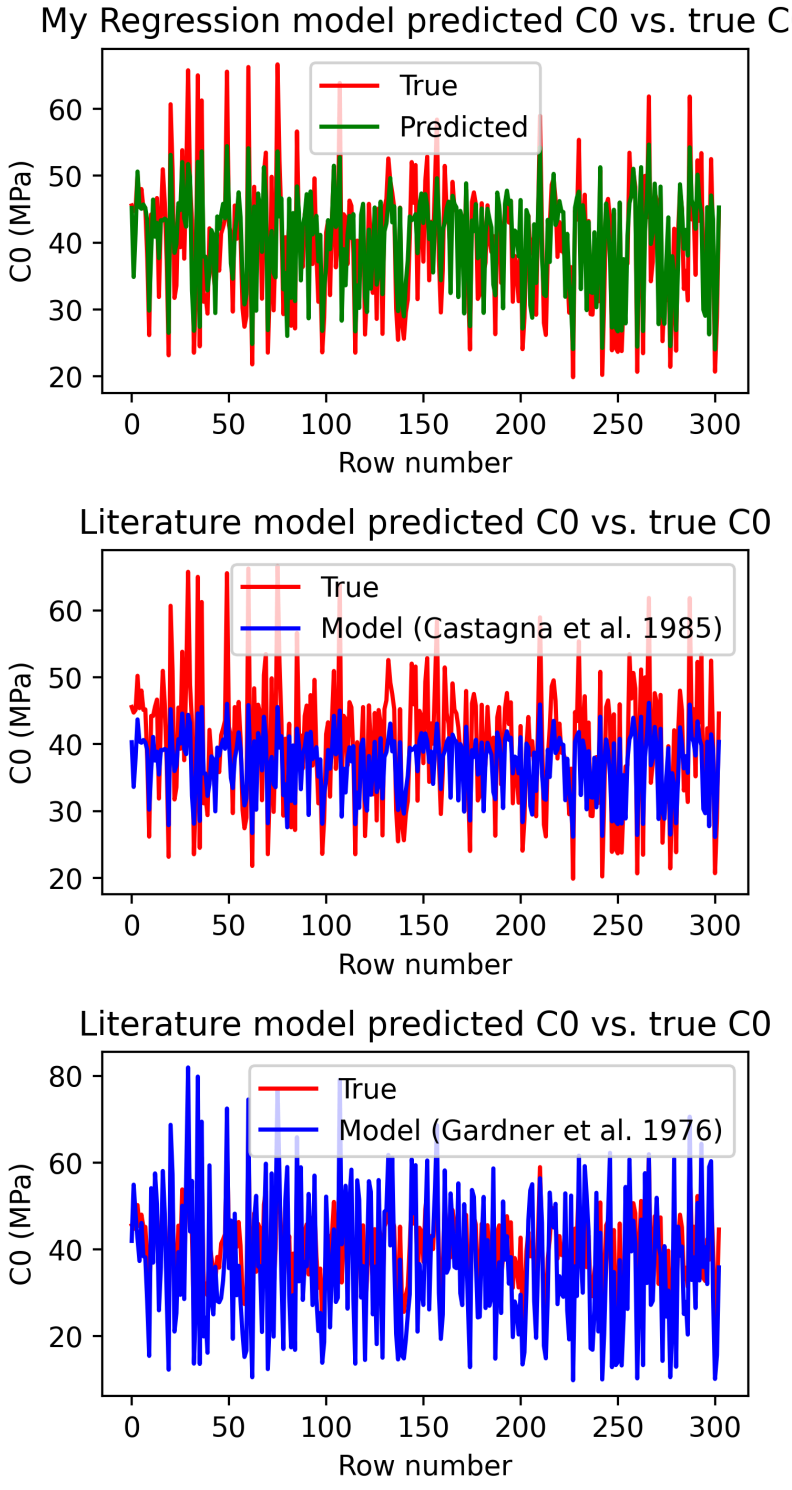
We now use our function on all our features. Based on MSE, RMSE, R^2 and visualization we conclude that the NEU feature is the best for predicting Vp.

In [32]:
```
def castagna():
    y = df["Vp"]
    X = df[["Vs"]]
    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.3, rand
    Co_Castagna = np.array(0.77*(1.16*x_test + 1.36)**2.93)
    return Co_Castagna
```

```
Co_Castagna = castagna()

def gardner():
    y = df["Vp"]
    X = df[["DEN"]]
    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.3, rand
    return np.array(0.77*(0.0003048*(x_test/0.23)**4)**2.93)

Co_gardner = gardner()

def Co(X,y, text):
    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.3, rand
    regression = LinearRegression()
    regression.fit(x_train, y_train)
    y_pred = regression.predict(x_test)

    Co_true = np.array(0.77*y_test**2.93)
    Co_my_model = 0.77*y_pred**2.93
    print(f'{text} R2=', r2_score(Co_true, Co_my_model))
    fig, axes = plt.subplots(3, 1, figsize=(4,8), dpi = 380)

    fig.suptitle(f"{text} Uniaxial Compressive Strength (C0) Comparison")
    # Plotting regression model predicted C0 and true C0
    axes[0].plot(Co_true, label = "True", color = "red")
    axes[0].plot(Co_my_model, label = "Predicted", color = "green")
    axes[0].set_xlabel("Row number")
    axes[0].set_ylabel("C0 (MPa)")
    axes[0].set_title("My Regression model predicted C0 vs. true C0")
    axes[0].legend()


    # Plotting literature model (Castagna et al. (1985)) predicted C0 and true C
    axes[1].plot(Co_true, label = "True", color = "red")
    axes[1].plot(Co_Castagna, label = "Model (Castagna et al. 1985)", color = "b
    axes[1].set_xlabel("Row number")
    axes[1].set_ylabel("C0 (MPa)")
    axes[1].set_title("Literature model predicted C0 vs. true C0")
    axes[1].legend()

    axes[2].plot(Co_true, label = "True", color = "red")
    axes[2].plot(Co_gardner, label = "Model (Gardner et al. 1976)", color = "blu
    axes[2].set_xlabel("Row number")
    axes[2].set_ylabel("C0 (MPa)")
    axes[2].set_title("Literature model predicted C0 vs. true C0")
    axes[2].legend()
    plt.tight_layout()
    plt.show()

y = df["Vp"]
X = df[["Vs"]]
Co(X, y, "Model 1: Vp vs (Vs)")

X = df[["DEN"]]
Co(X,y, "Model 2: Vp vs (DEN)")

X = df[["NEU"]]
Co(X,y, "Model 3: Vp vs (NEU)")
```

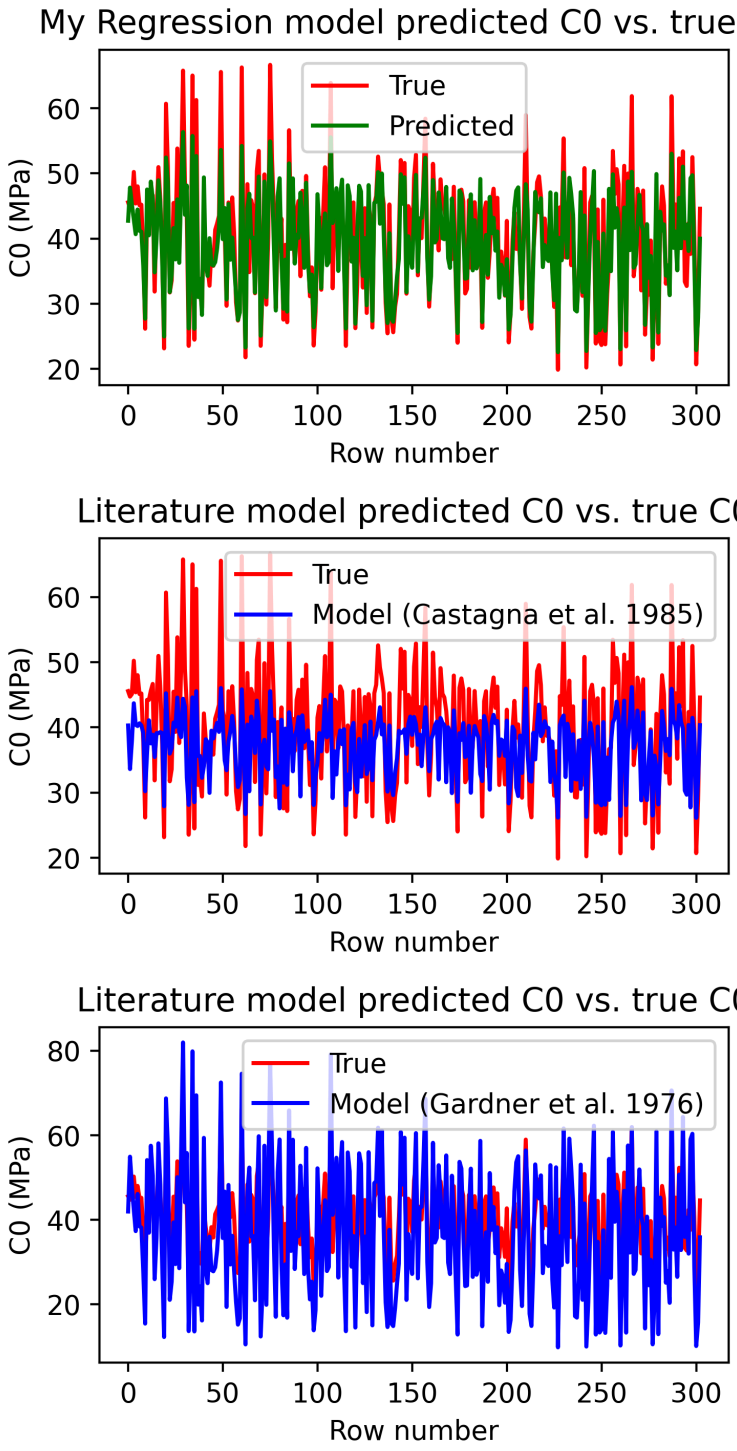
```
Model 1: Vp vs (Vs) R2= 0.6842398643508283
```

Model 1: Vp vs (Vs) Uniaxial Compressive Strength (C0) Comparison



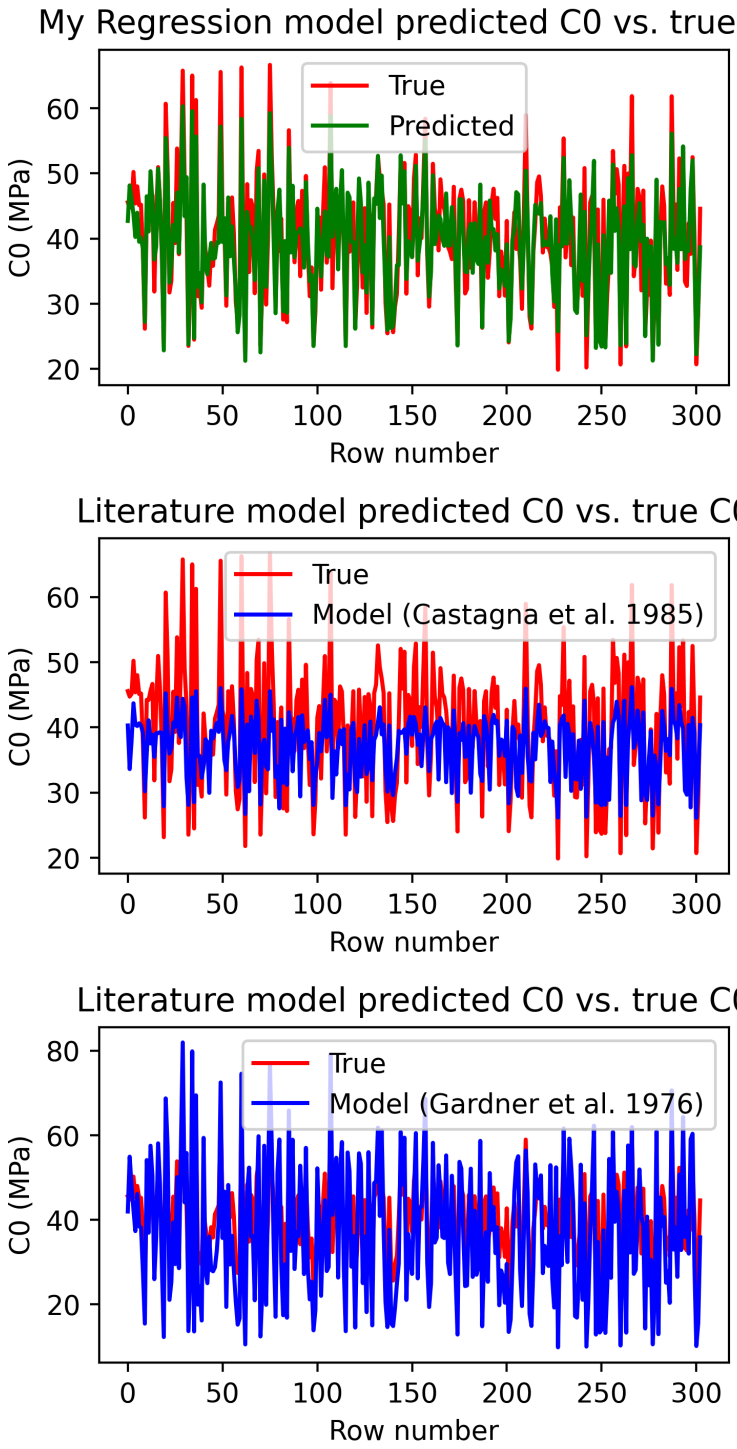

```
Model 2: Vp vs (DEN) R2= 0.7445100019197923
```

Model 2: Vp vs (DEN) Uniaxial Compressive Strength (C0) Comparison




```
Model 3: Vp vs (NEU) R2= 0.8088794919836573
```

Model 3: Vp vs (NEU) Uniaxial Compressive Strength (C0) Comparison



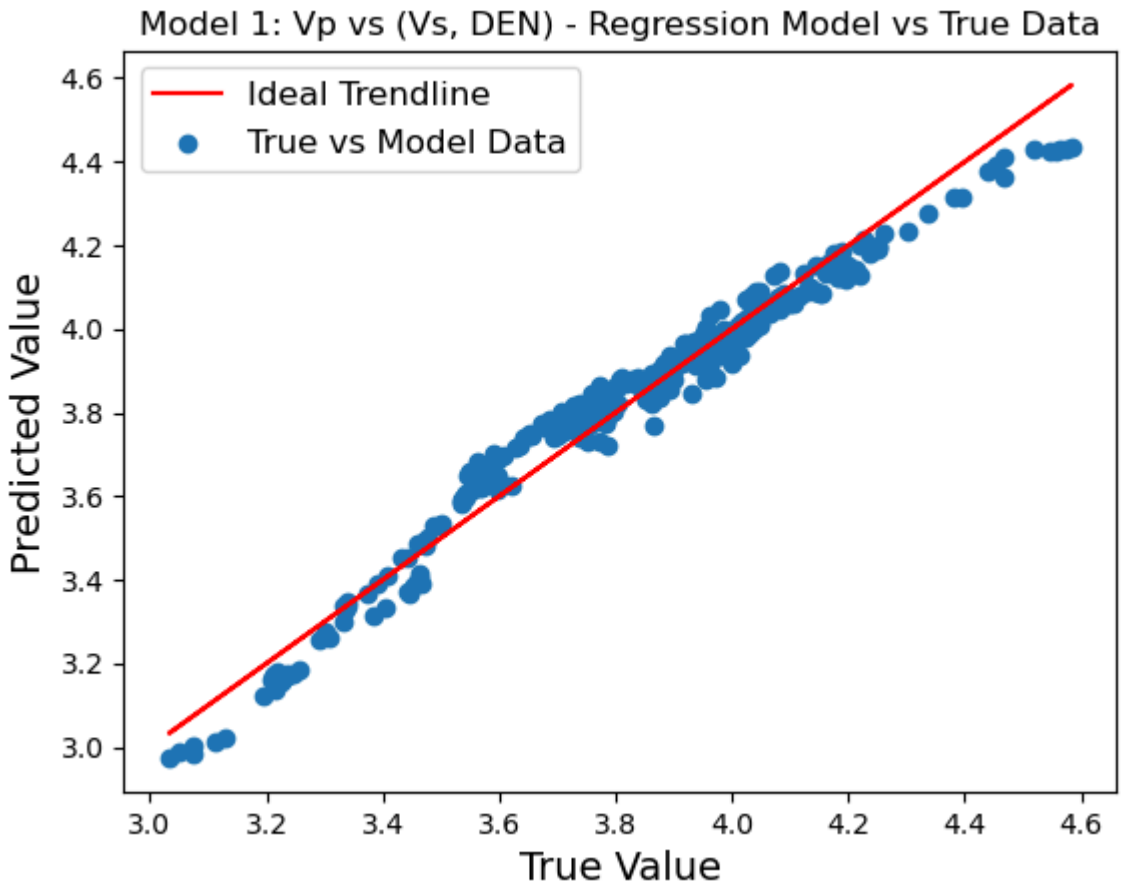
We have created functions for calculating the Uniaxial Compressive strength(Co) of Castagna, Gardner and our regression models for all features and then displaying the models for comparison. We also print the R^2 score to evaluate how close our model's Co is to the true Co. From the plots we see that the Gardner model is the closest, the Castagna model is the worst and that our model with NEU is second best.

Multivariable regression

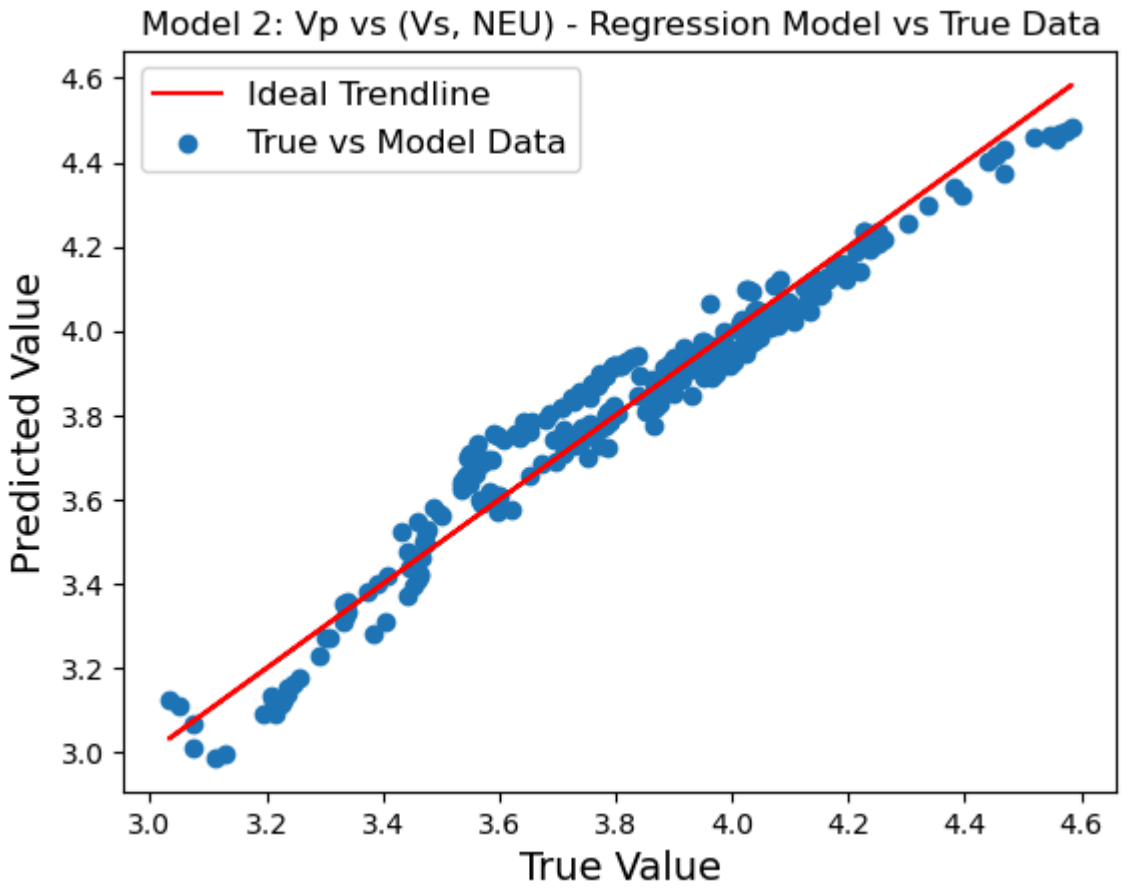
In [33]:
```
y = df['Vp']

X = df[['Vs', 'DEN']]
```

```
Reg(X, y, "Model 1: Vp vs (Vs, DEN)")

X = df[['Vs', 'NEU']]
Reg(X, y, "Model 2: Vp vs (Vs, NEU)")

X = df[['DEN', 'NEU']]
Reg(X, y, "Model 3: Vp vs (DEN, NEU)")

X = df[['Vs', 'DEN', 'NEU']]
Reg(X, y, "Model 4: Vp vs (Vs, DEN, NEU)")
```

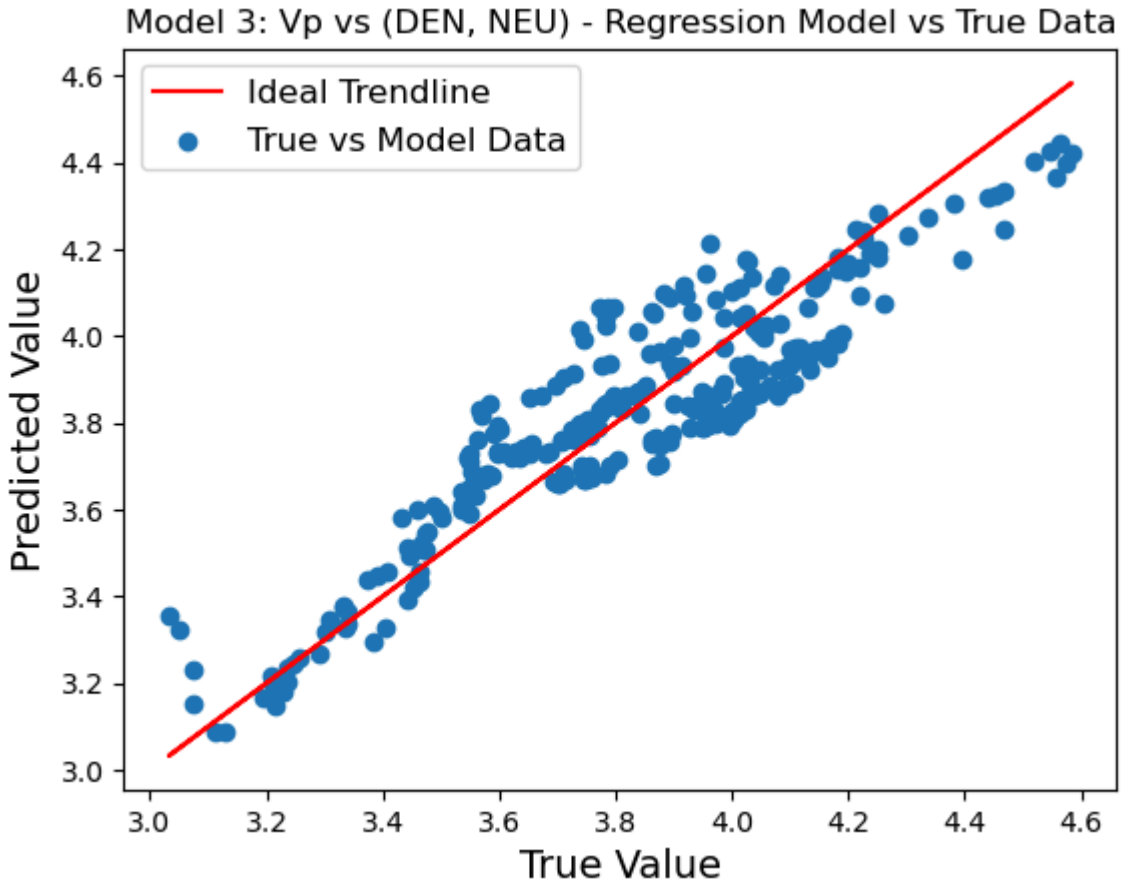
```
[1.09977327 1.93553889] -3.081868294200706
Model 1: Vp vs (Vs, DEN)
Mean Squared Error: 0.002975878622024917
Root Mean Squared Error: 0.054551614293482806
R-Square: 0.9695031510787857
Adjusted R-Square: 0.9692998387526442
```



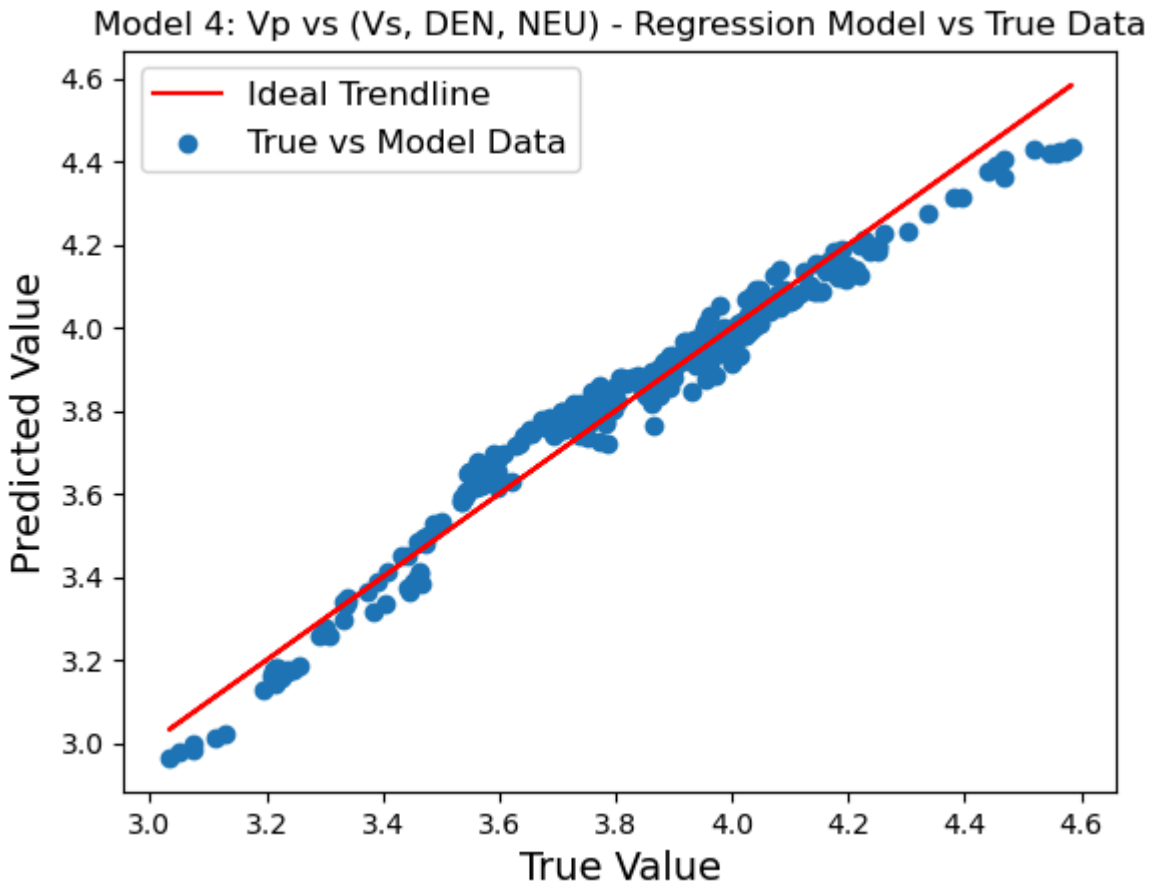

```
[ 0.9684806  -3.71943116] 2.643017996345125
Model 2: Vp vs (Vs, NEU)
Mean Squared Error: 0.00478381368896647
Root Mean Squared Error: 0.06916511901939063
R-Square: 0.9509754052937897
Adjusted R-Square: 0.9506485746624149
```




```
[-0.66528339 -6.42794878] 6.805850414550262
Model 3: Vp vs (DEN, NEU)
Mean Squared Error: 0.01659739126733271
Root Mean Squared Error: 0.12883086302331717
R-Square: 0.8299096844139066
Adjusted R-Square: 0.828775748976666
```



```
[1.11055712 2.058404   0.24853289] -3.45294807480616
Model 4: Vp vs (Vs, DEN, NEU)
Mean Squared Error: 0.00295606848381938
Root Mean Squared Error: 0.05436973867712976
R-Square: 0.9697061656733633
Adjusted R-Square: 0.9694022141583803
```

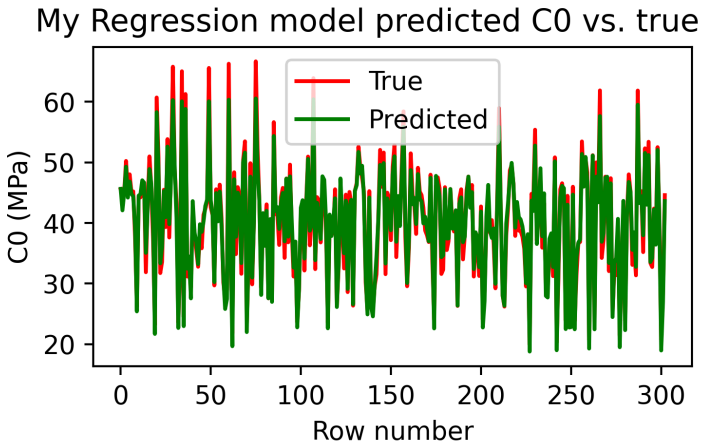


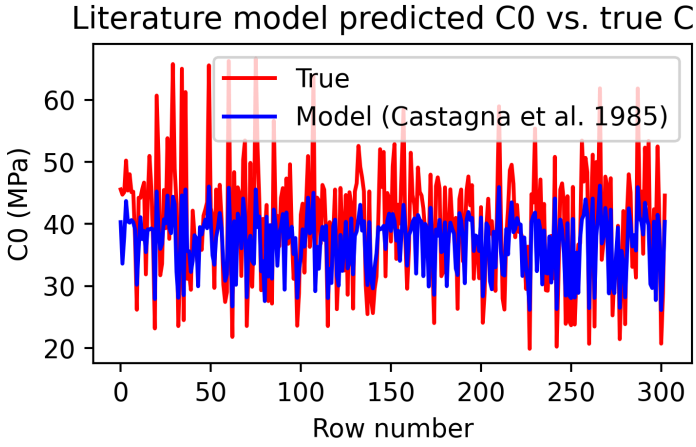
We now use our function again, but this time for multivariable regression. We see imediately that the multivariable regression is much better at predicting Vp and that using all the features is the best but it is very close between using (Vs, DEN) and (Vs, DEN, NEU).

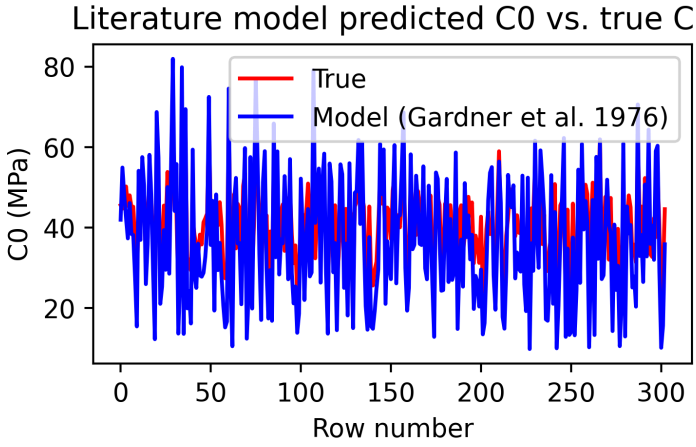
In [34]:
```
y = df["Vp"]

X = df[['Vs', 'DEN']]
Co(X, y, "Model 1: Vp vs (Vs, DEN)")

X = df[['Vs', 'NEU']]
Co(X, y, "Model 2: Vp vs (Vs, NEU)")

X = df[['DEN', 'NEU']]
Co(X, y, "Model 3: Vp vs (DEN, NEU)")

X = df[['Vs', 'DEN', 'NEU']]
Co(X, y, "Model 4: Vp vs (Vs, DEN, NEU)")
```

```
Model 1: Vp vs (Vs, DEN) R2= 0.9674000289939834
```

Model 1: Vp vs (Vs, DEN) Uniaxial Compressive Strength (C0) Comparison

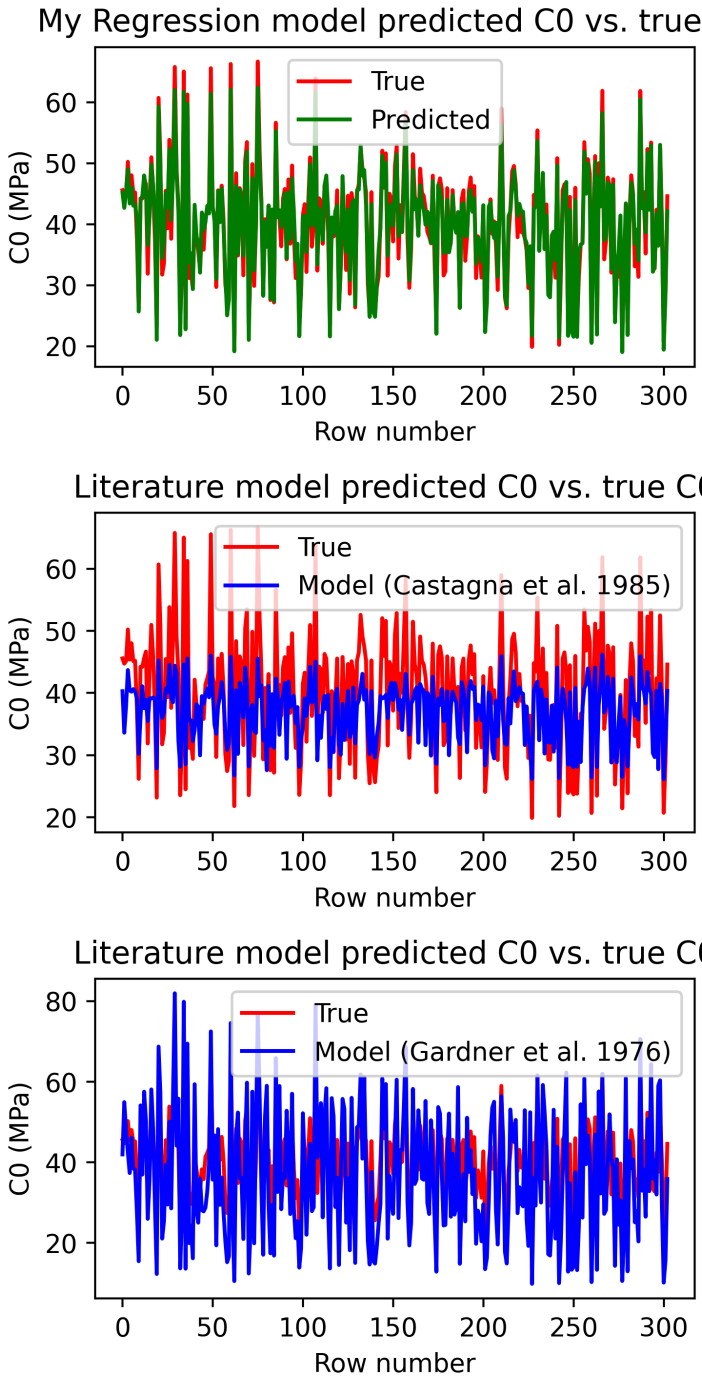






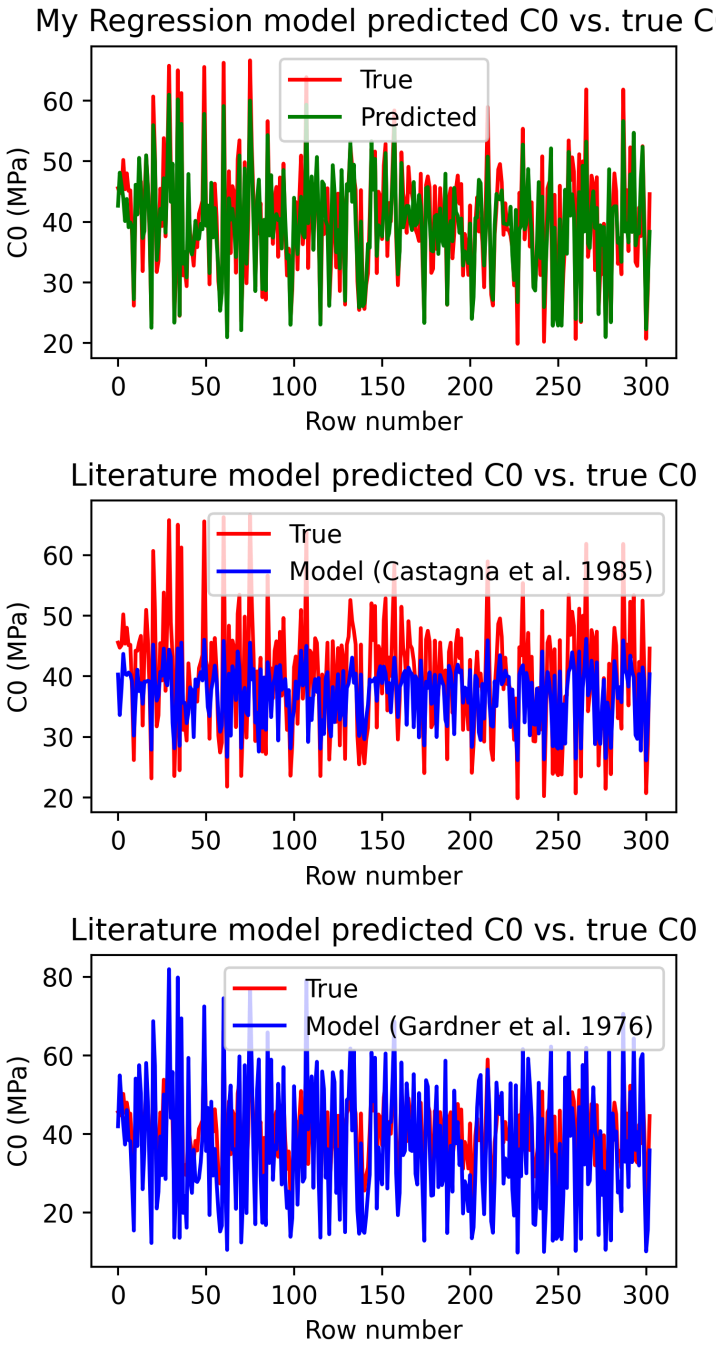

```
Model 2: Vp vs (Vs, NEU) R2= 0.9527327828953167
```

Model 2: Vp vs (Vs, NEU) Uniaxial Compressive Strength (C0) Comparison



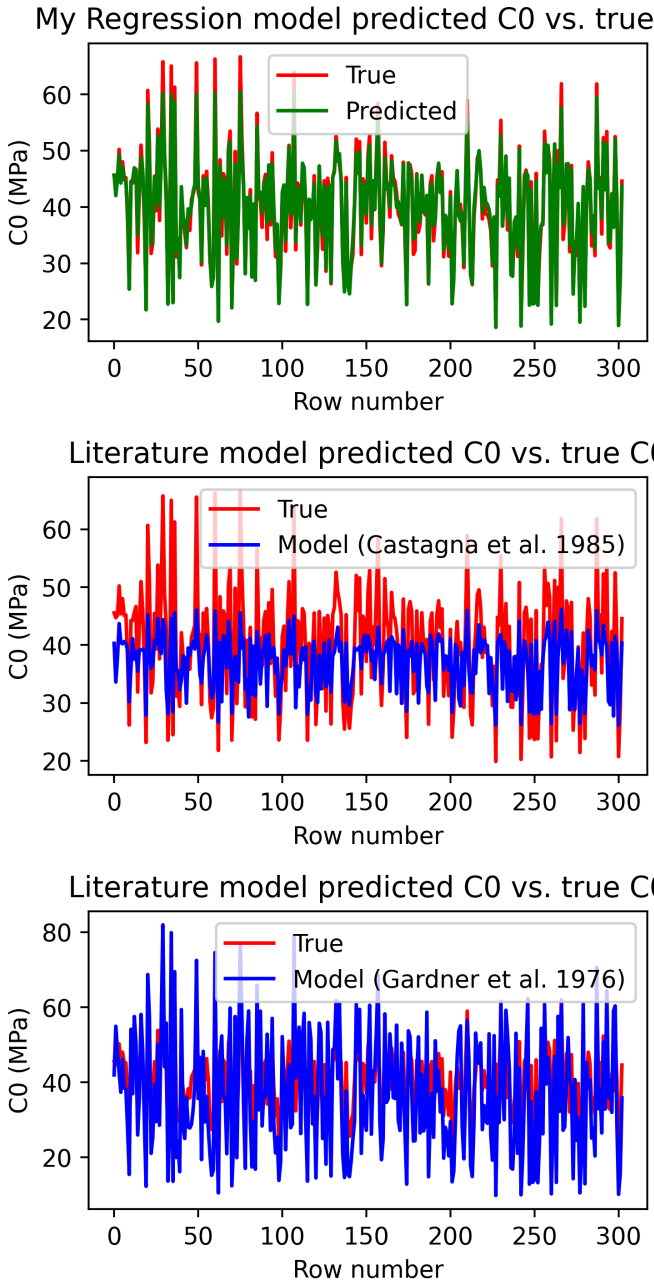

```
Model 3: Vp vs (DEN, NEU) R2= 0.8126519432680854
```

Model 3: Vp vs (DEN, NEU) Uniaxial Compressive Strength (C0) Comparison




```
Model 4: Vp vs (Vs, DEN, NEU) R2= 0.9672937918961243
```

Model 4: Vp vs (Vs, DEN, NEU) Uniaxial Compressive Strength (C0) Comparison



We now calculate Co for our multivariable models and compare them against both Castagna and Gardner. We see clear difference between our Linear and multivariable regression models, multivariable is clearly more accurate. We also see that the regression with all features is best and it is very close to the Gardner model, so close that we can't really tell which is superior.

Conclusion

Multivariable regression proved superior to linear regression in estimating Vp from log data, showing how important it is to incorporate multiple features for accurate predictions. The results underscore that, in UCS estimation, combining Vs, DEN and NEU yields a more accurate prediction than using individual features. This task confirmed the value of multivariable modeling in enhancing prediction accuracy for applications in wellbore stability analysis.

Reflections

Though building and evaluating both linear and multivariable regression models, we learned how important it can be to combine features in a multivariable model to improve the accuracy of the prediction. Linear regression was also important in showing the individual predictive power of each feature, but it was not as effective as multivariable. We also learned more about using plots to visualize differences in regression models and Uniaxial compressive strength.

References

[1]: Geek for Geeks, 10.11.2024, https://www.geeksforgeeks.org/regression-in-machine-learning/

[2]: Corporate Finance Institute, 10.11.2024, https://corporatefinanceinstitute.com/resources/data-science/r-squared/