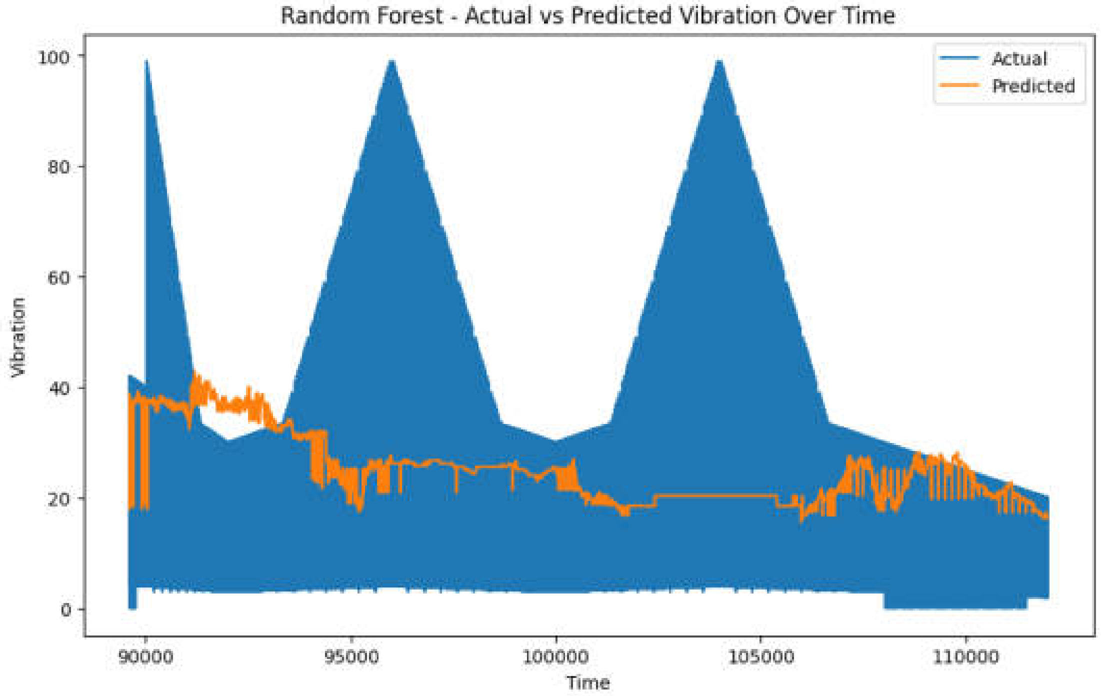
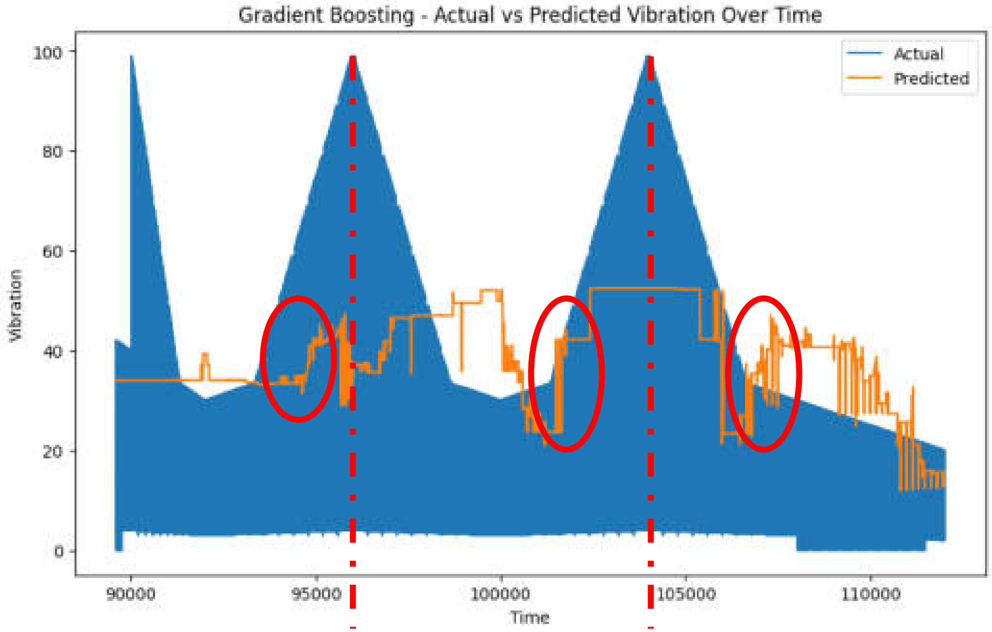
Conclusion: Optimized

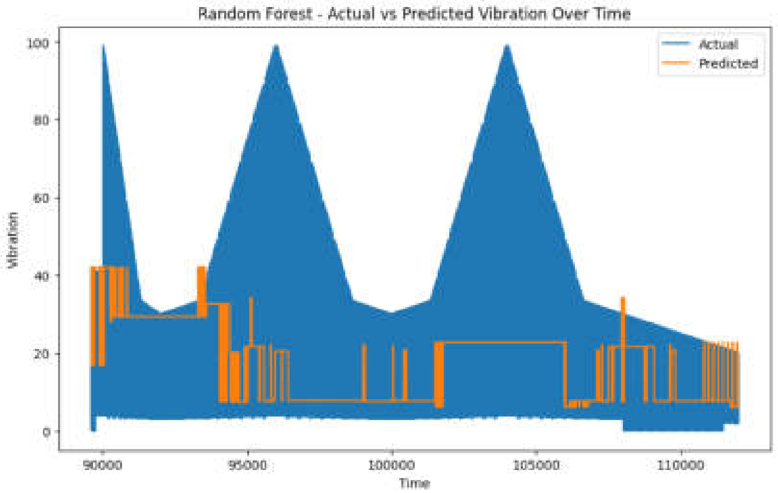
- 80 Estimators for forest
- 300 for gradient boosting
- Run time: 2m 33 seconds

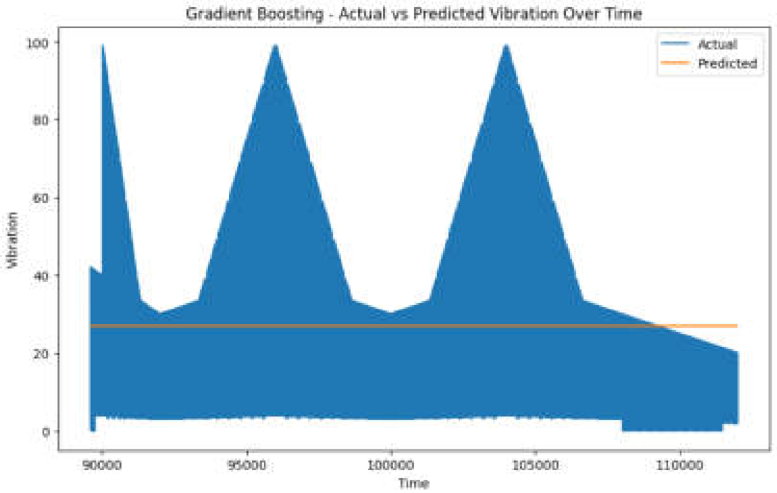


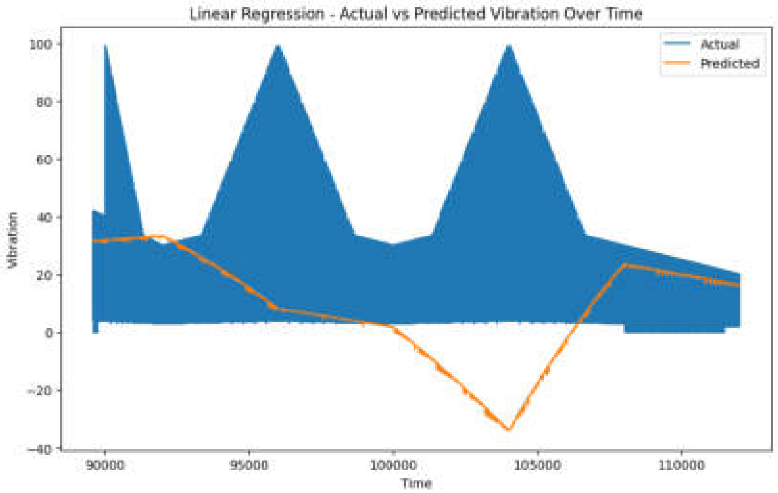
- Inspections signaling shown best by the gradient boosting,shows the change the clearest



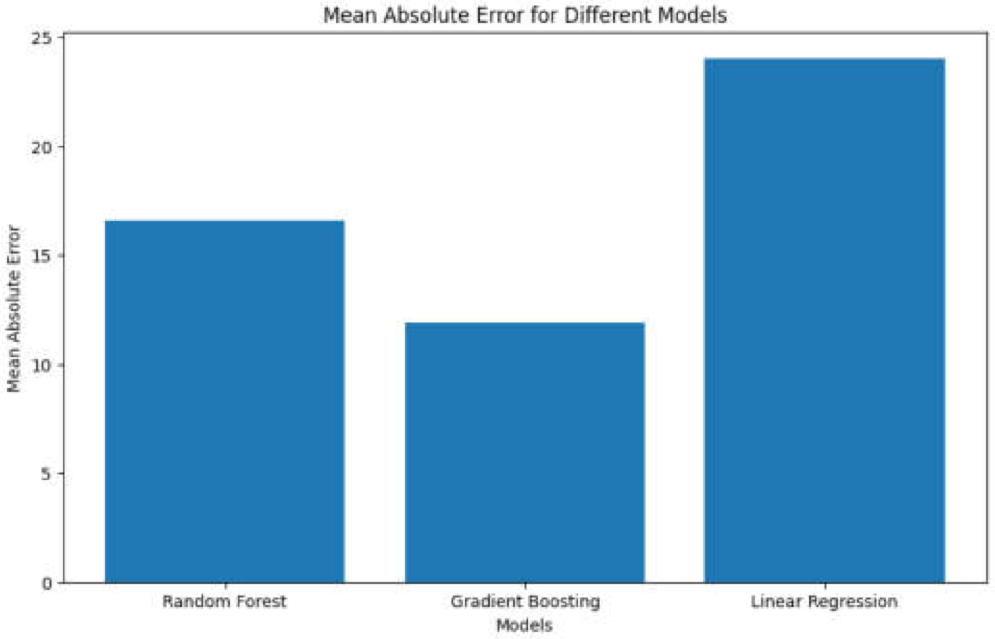
Random Forest regressor, Gradient Boosting Regressor Linear Regression



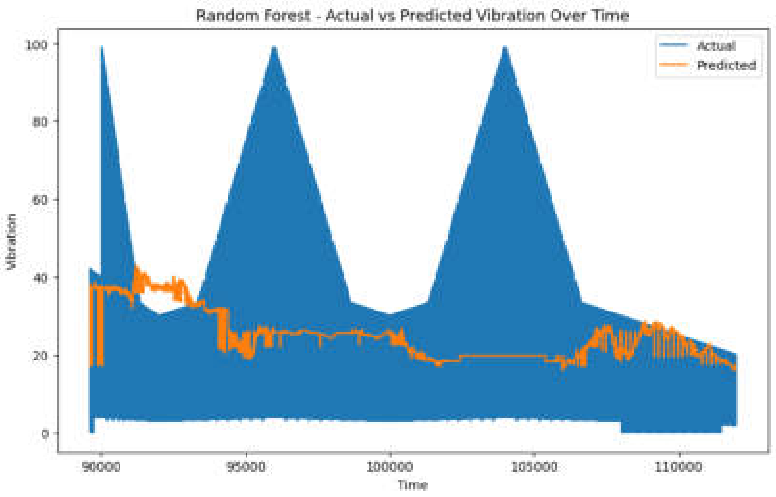


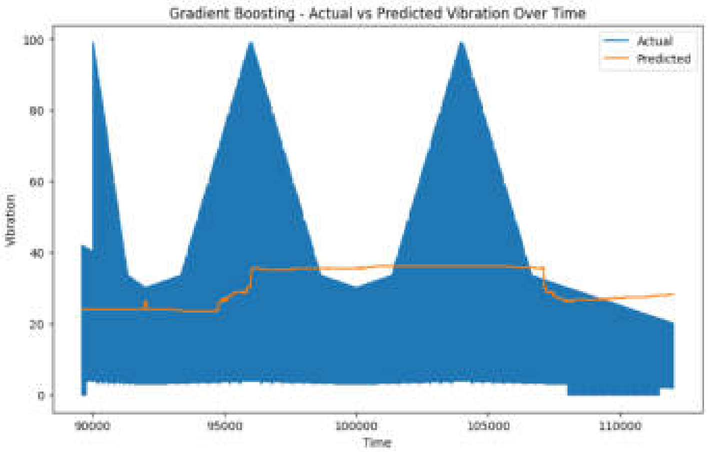


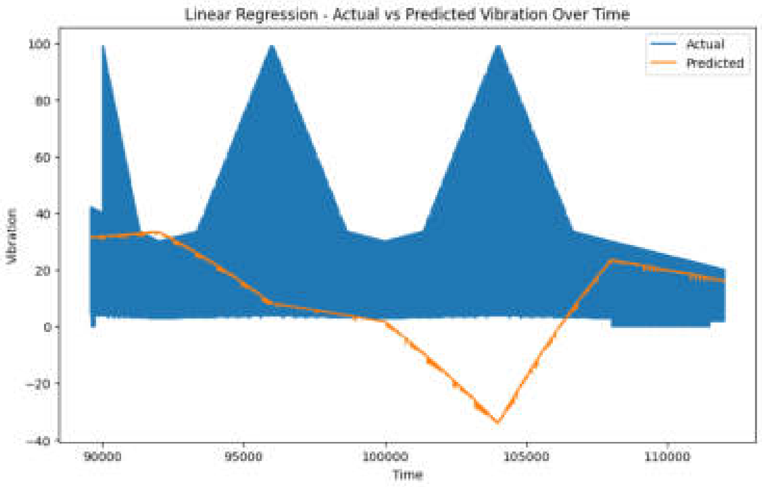
3 Estimators, run time: 6seconds



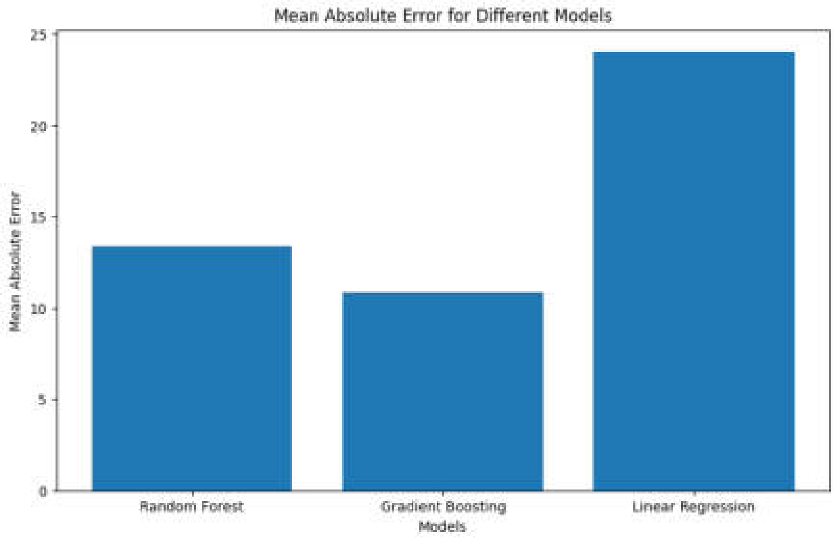
Random Forest regressor, Gradient bOosting Regressor Linear Regression



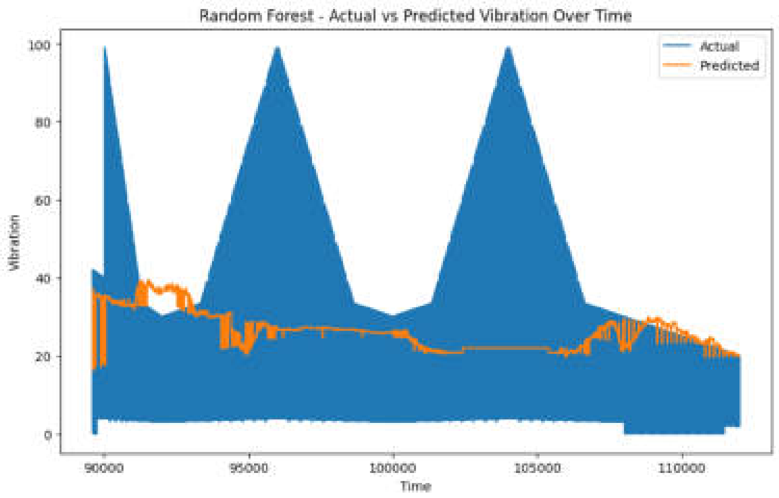


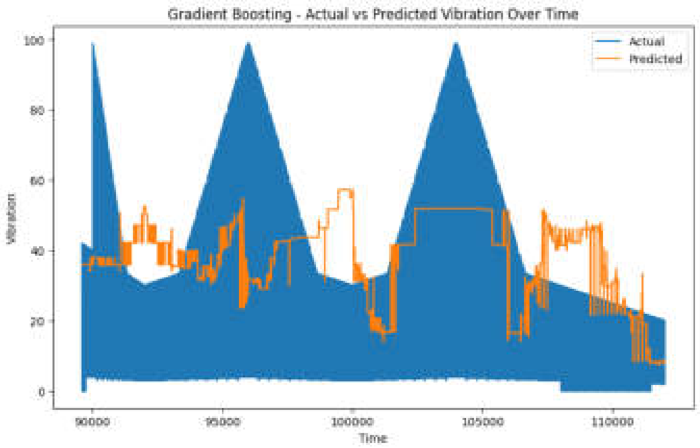


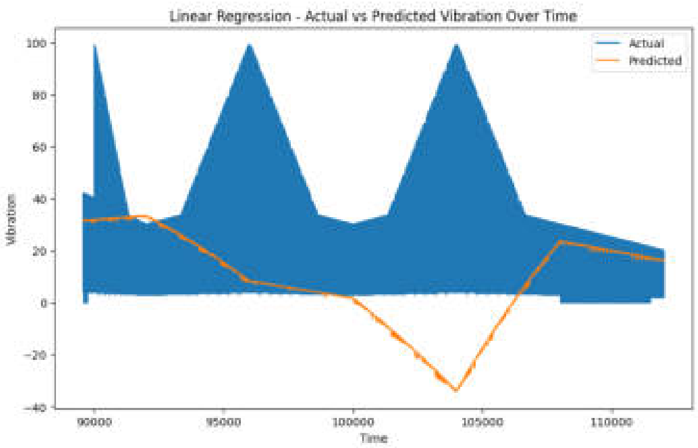
100 Estimators, run time: 1m 54 seconds



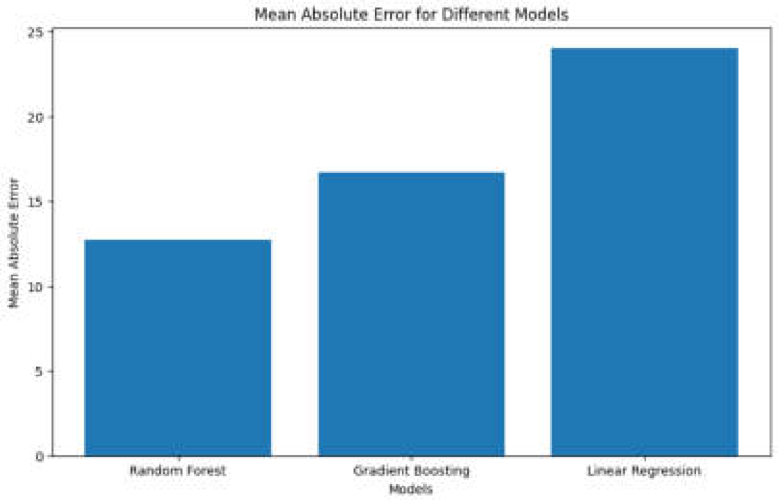
Random Forest regressor, Gradient bOosting Regressor Linear Regression



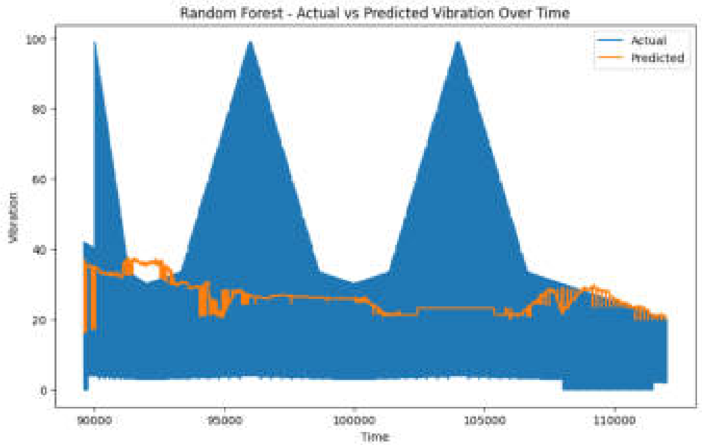


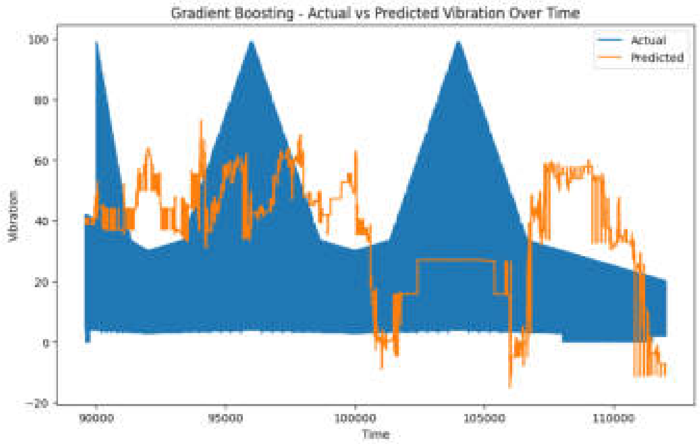


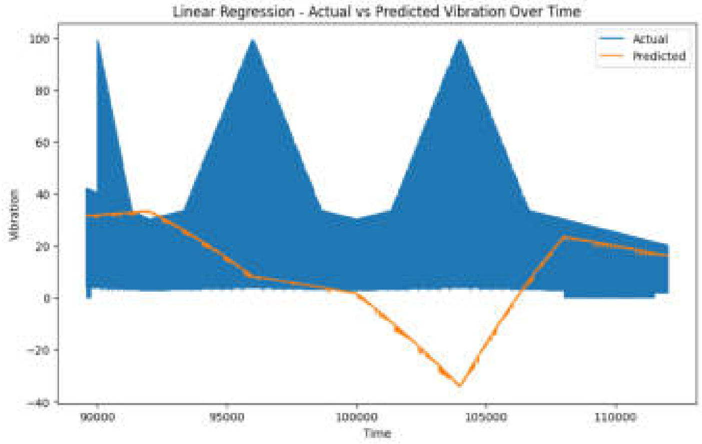
400 Estimators, run time: 7m 37 seconds



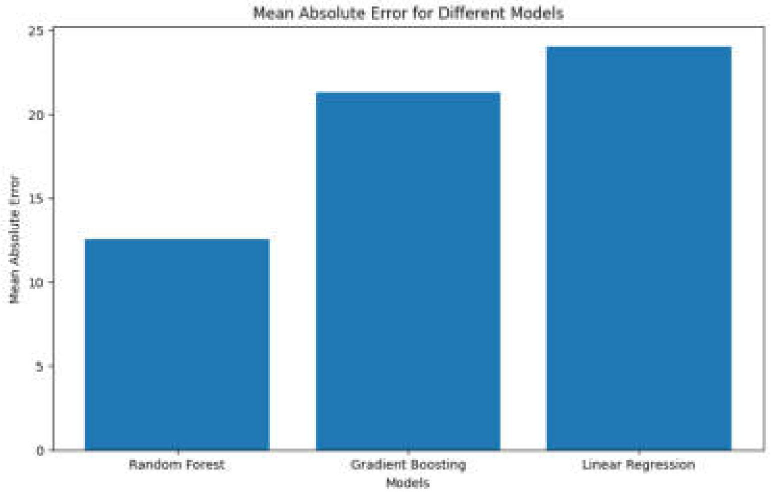
Random Forest regressor, Gradient bOosting Regressor Linear Regression



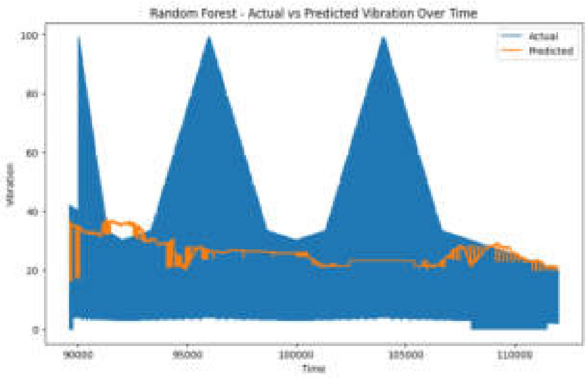


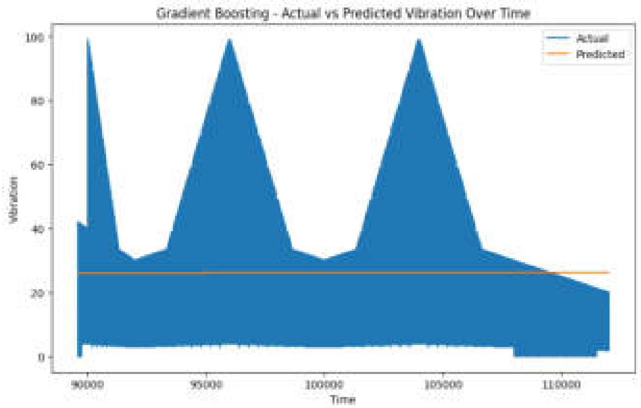


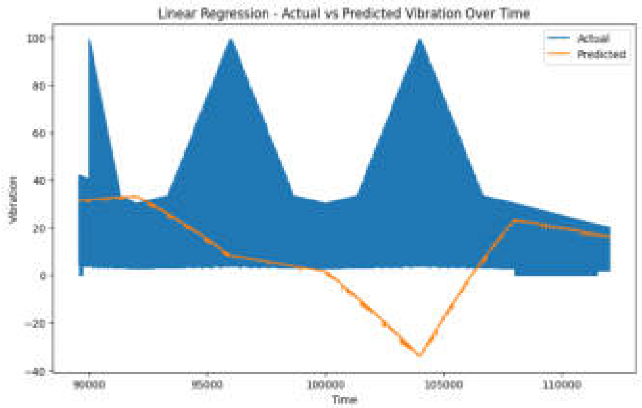
1000 Estimators, run time: 18m 45 seconds



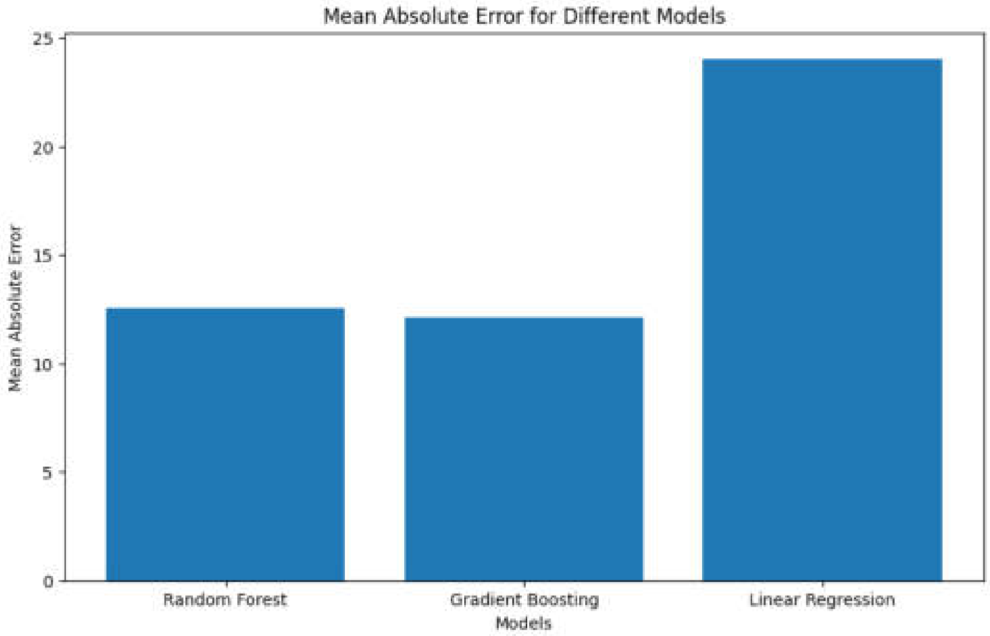
Random Forest regressor, Gradient boosting Regressor Linear Regression







1300 Estimators for forest, run time: 18m 38 seconds



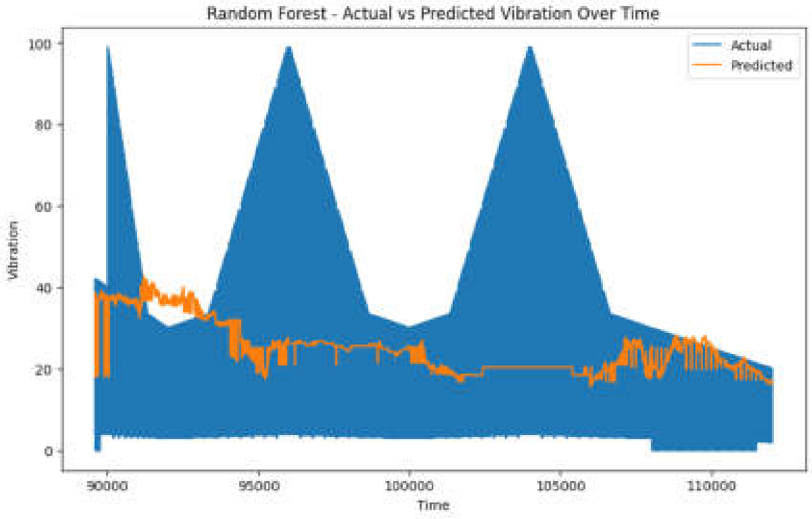
Conclusion

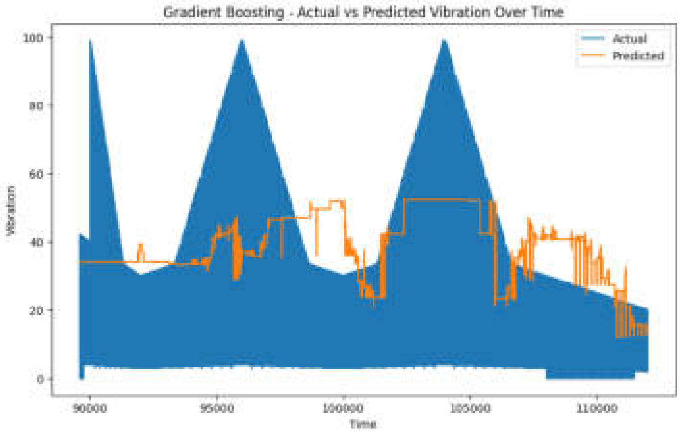
•For Random Forest, the MAE decreases as the number of estimators increases. The error seems to stabilize at around 13 after reaching 100 estimators. Further increasing the number of estimators does not significantly reduce the error.

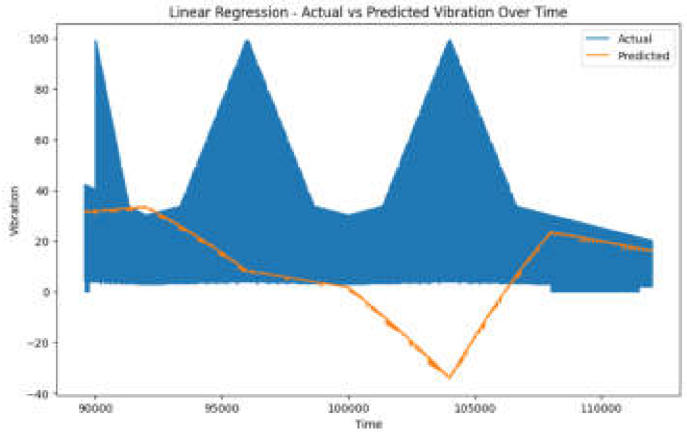
•For Gradient Boosting, the MAE decreases initially and then stabilizes around 13 after reaching 11 estimators. Additional estimators do not contribute much to error reduction.

•For Linear Regression, the MAE remains constant at 24 regardless of the number of estimators.

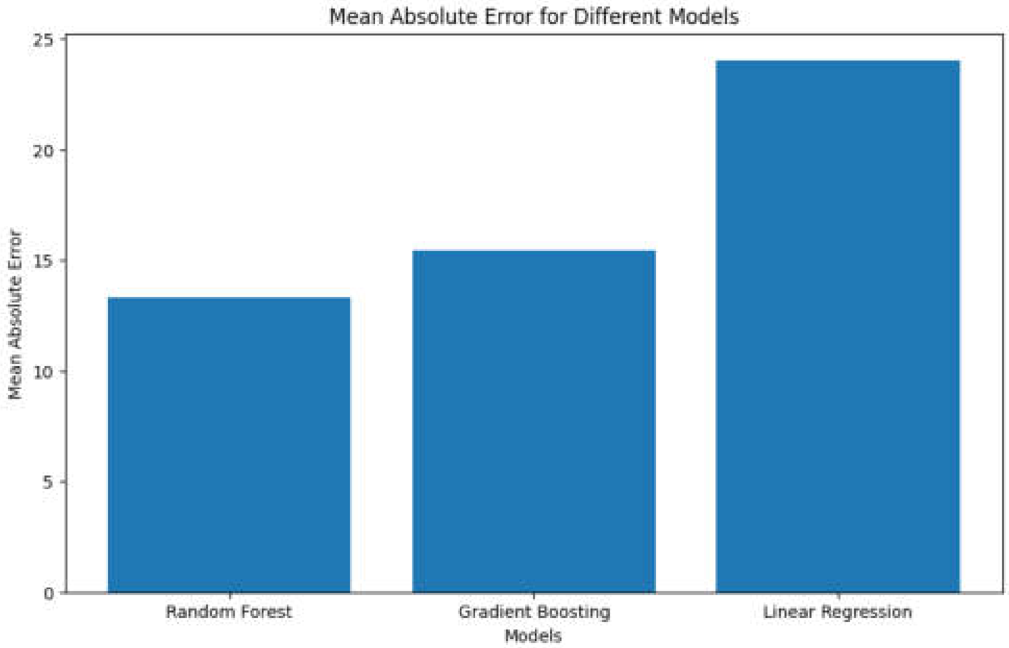
Random Forest regressor, Gradient bOosting Regressor Linear Regression





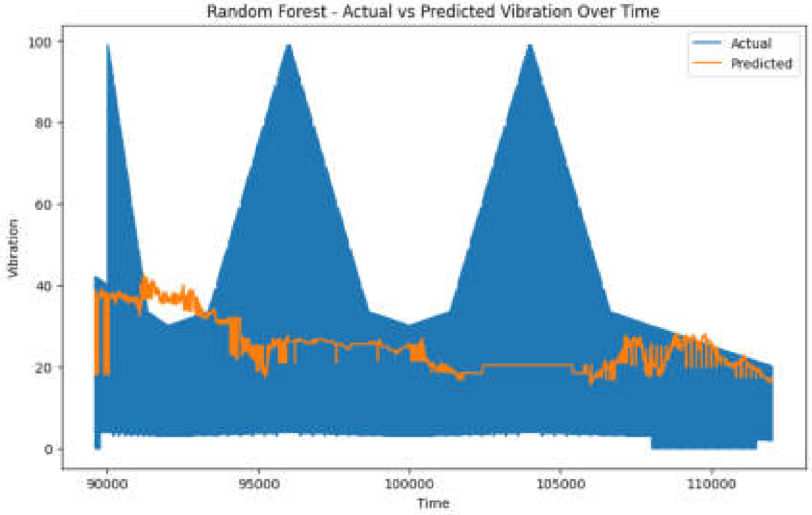


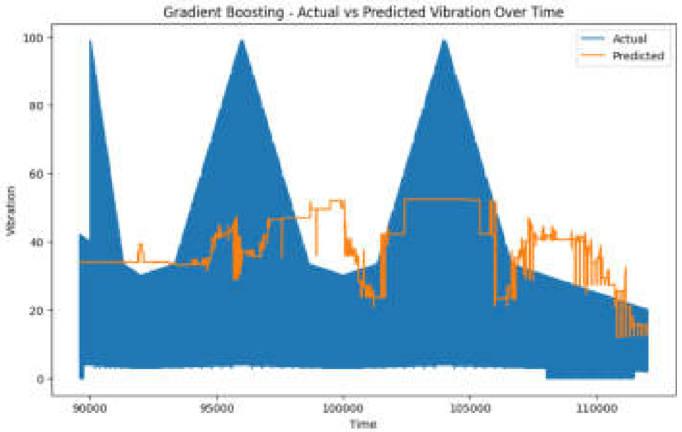
80 Estimators for forest,300 for gradient boosting run time: 2m 33 seconds

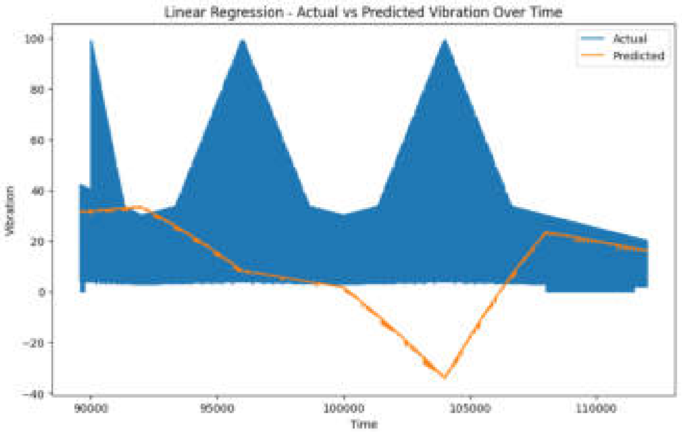


Time Is Money, Optimize to reduce run time too!

Random Forest regressor, Gradient bOosting Regressor Linear Regression

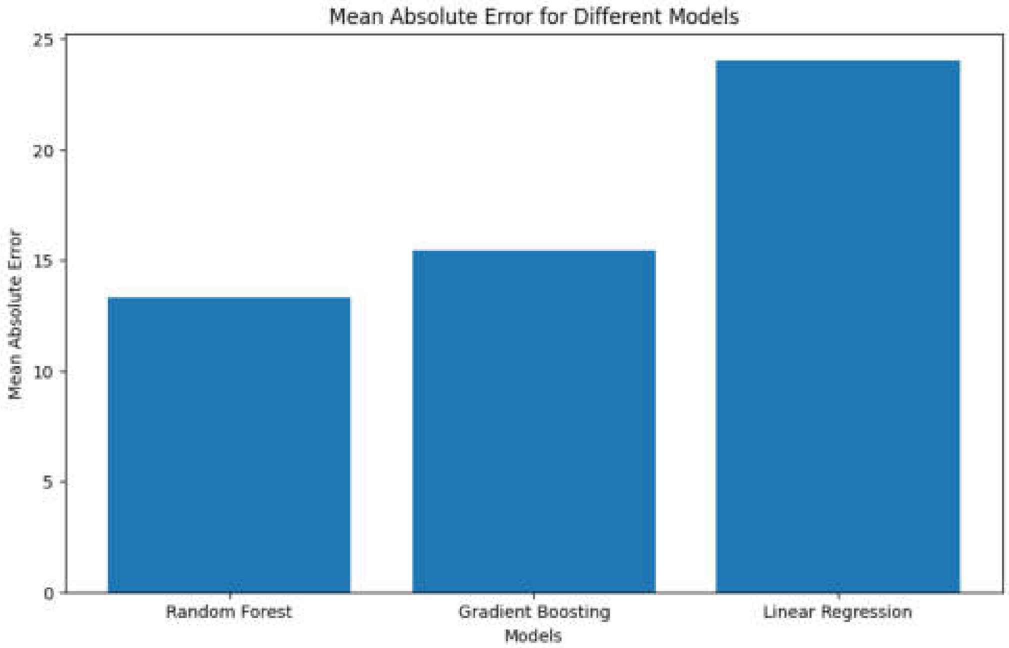






80 Estimators for forest,300 for gradient boosting run time:

2m 33 seconds



Final code revision

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv('predictive-maintenance-dataset.csv')

# Replace NaN values with zero in the 'vibration' column
df['vibration'].fillna(0, inplace=True)

# Separate the target (y) and input (X) variables
X = df.drop('vibration', axis=1)
y = df['vibration']

# Split the data into training and test sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Define the models
models = [
('Random Forest', RandomForestRegressor(n_estimators=80, random_state=0)),
('Gradient Boosting', GradientBoostingRegressor(n_estimators=300, random_state=0)),
('Linear Regression', LinearRegression())
]

# Train and predict with each model
mae_values = []
for model_name, model in models:
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Calculate the mean absolute error
mae = mean_absolute_error(y_test, y_pred)
mae_values.append(mae)
print(f'{model_name} - Mean Absolute Error: {mae}')

# Find the index where vibration exceeds the threshold
threshold = 80  # Set the threshold for repair/inspection
exceed_threshold_indices = np.where(y_pred >= threshold)[0]

# Plot actual vs predicted values
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(X_test.index, y_test, label='Actual')
ax.plot(X_test.index, y_pred, label='Predicted')

ax.set_xlabel('Time')
ax.set_ylabel('Vibration')
ax.set_title(f'{model_name} - Actual vs Predicted Vibration Over Time')
ax.legend()
plt.show()

# Plot the mean absolute error for each model
plt.figure(figsize=(10, 6))
models_names = [model_name for model_name, _ in models]
plt.bar(models_names, mae_values)
plt.xlabel('Models')
plt.ylabel('Mean Absolute Error')
plt.title('Mean Absolute Error for Different Models')
plt.show()
```