

```
from google.colab import files
uploaded = files.upload()
```

Choose Files | No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# 1. LOAD DATASET
# Replace with your actual file path
df = pd.read_excel("waam_dataset.xlsx")
```

```
print("Dataset Loaded Successfully!")
df.head()
```

Dataset Loaded Successfully!

	EXPERIMENTS	Voltage	Wire Feed Speed(WFS)	Travel Speed (mm/min)	CTWD (Working Distance)	Sz1	Sz2	Sz3	Sz4	Sz5	Sz6	Sz7	Sz8	Sz9	Sz10	Sz11	Sz12	Sz13
0	1	12.0	75	100	10.5	5.00	4.35	3.46	4.22	4.21	3.33	3.44	3.08	3.91	2.75	4.01	3.60	3.46
1	2	12.0	75	100	10.5	3.78	3.84	3.85	3.67	3.75	3.82	4.99	3.36	3.67	3.63	3.58	3.41	3.73
2	3	12.0	75	100	10.5	3.68	3.45	3.53	3.63	3.86	3.70	3.84	3.81	3.79	3.93	3.75	3.78	3.74
3	4	14.0	100	125	11.0	3.16	3.11	3.12	3.15	3.03	3.07	2.98	3.22	3.14	3.31	3.13	3.57	3.68
4	5	14.0	100	125	11.0	3.18	3.12	3.12	3.45	3.15	3.09	3.38	3.23	3.18	3.40	3.64	4.01	3.87

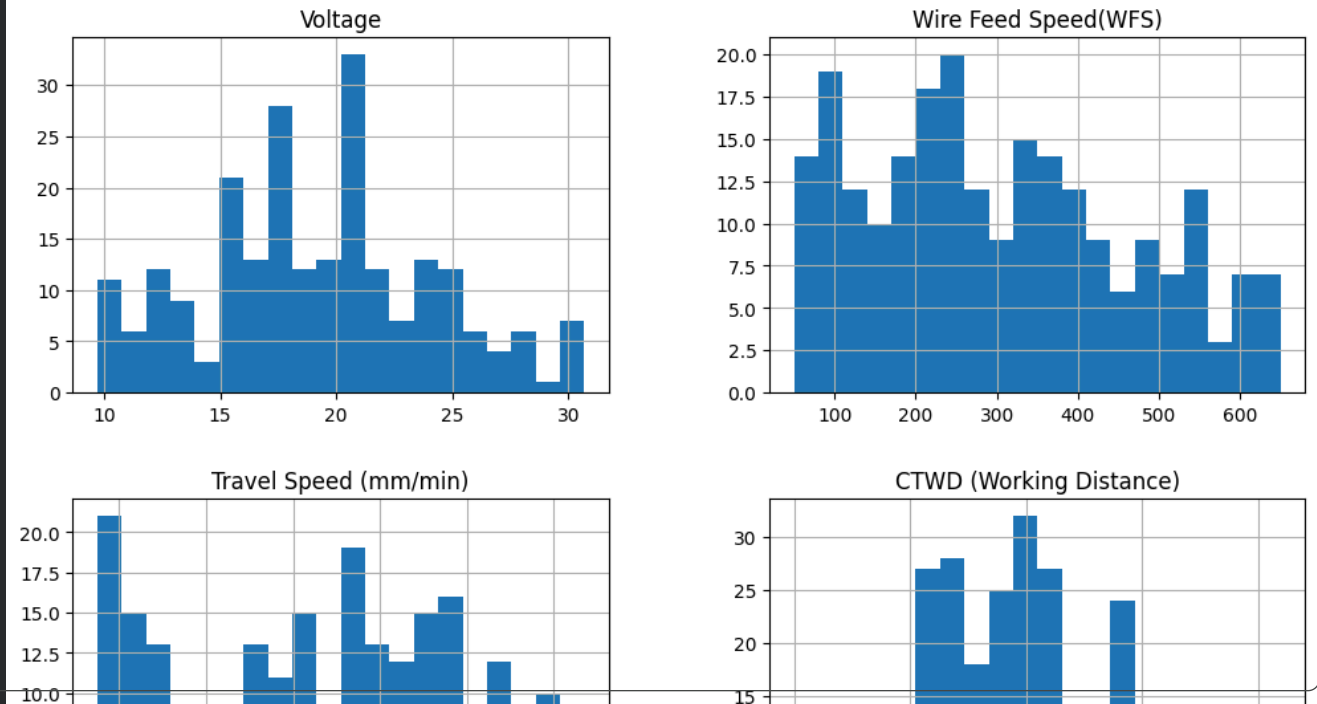
```
df.columns = df.columns.str.strip() # remove spaces
df.columns = df.columns.str.replace('\n','')
df.columns = df.columns.str.replace('\r','')
df.columns = df.columns.str.replace(' ','')
df.columns = df.columns.str.replace(u'\xa0','') # remove non-breaking spaces
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,8))
df[['Voltage','Wire Feed Speed(WFS)','Travel Speed (mm/min)','CTWD (Working Distance)']].hist(
    bins=20, figsize=(12,8))
)
plt.suptitle("Input Feature Distributions")
plt.show()
plt.savefig("figures/feature_histograms.png", dpi=300)
plt.close()
```

<Figure size 1200x800 with 0 Axes>

Input Feature Distributions

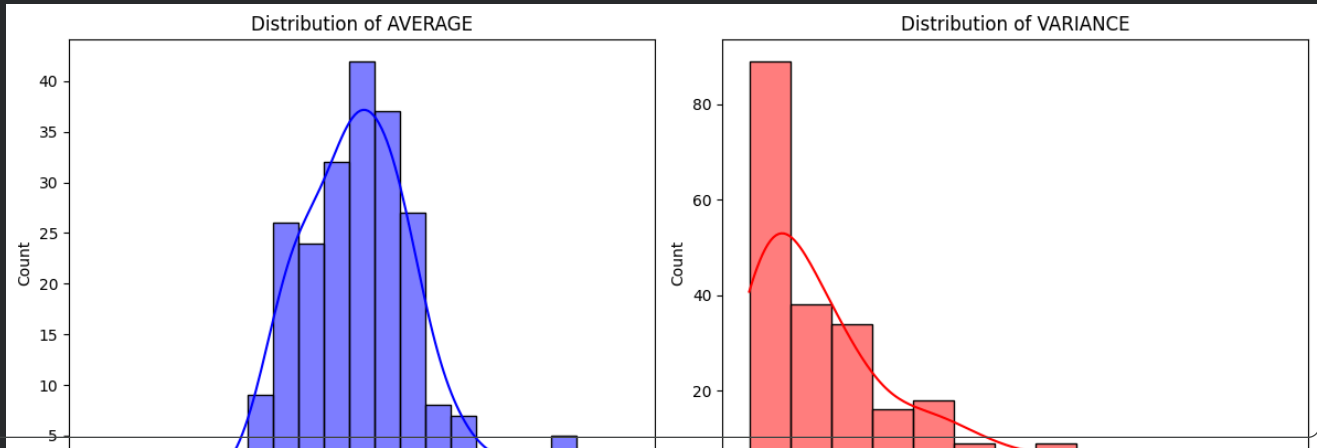


```
plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
sns.histplot(df['AVERAGE'], kde=True, color='blue')
plt.title("Distribution of AVERAGE")

plt.subplot(1,2,2)
sns.histplot(df['VARIANCE'], kde=True, color='red')
plt.title("Distribution of VARIANCE")

plt.tight_layout()
plt.show()
```



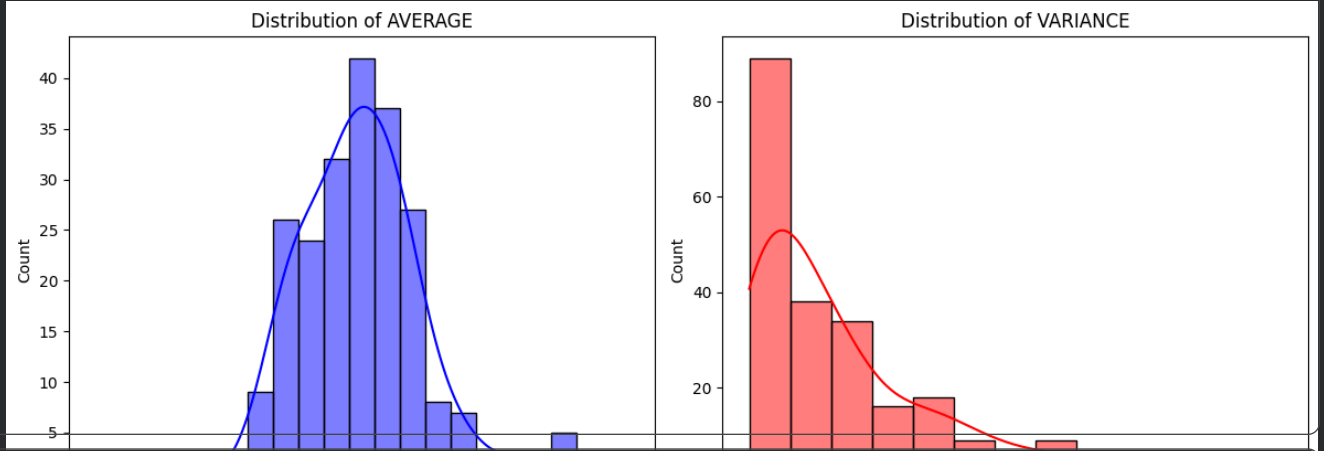
```
plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
sns.histplot(df['AVERAGE'], kde=True, color='blue')
plt.title("Distribution of AVERAGE")

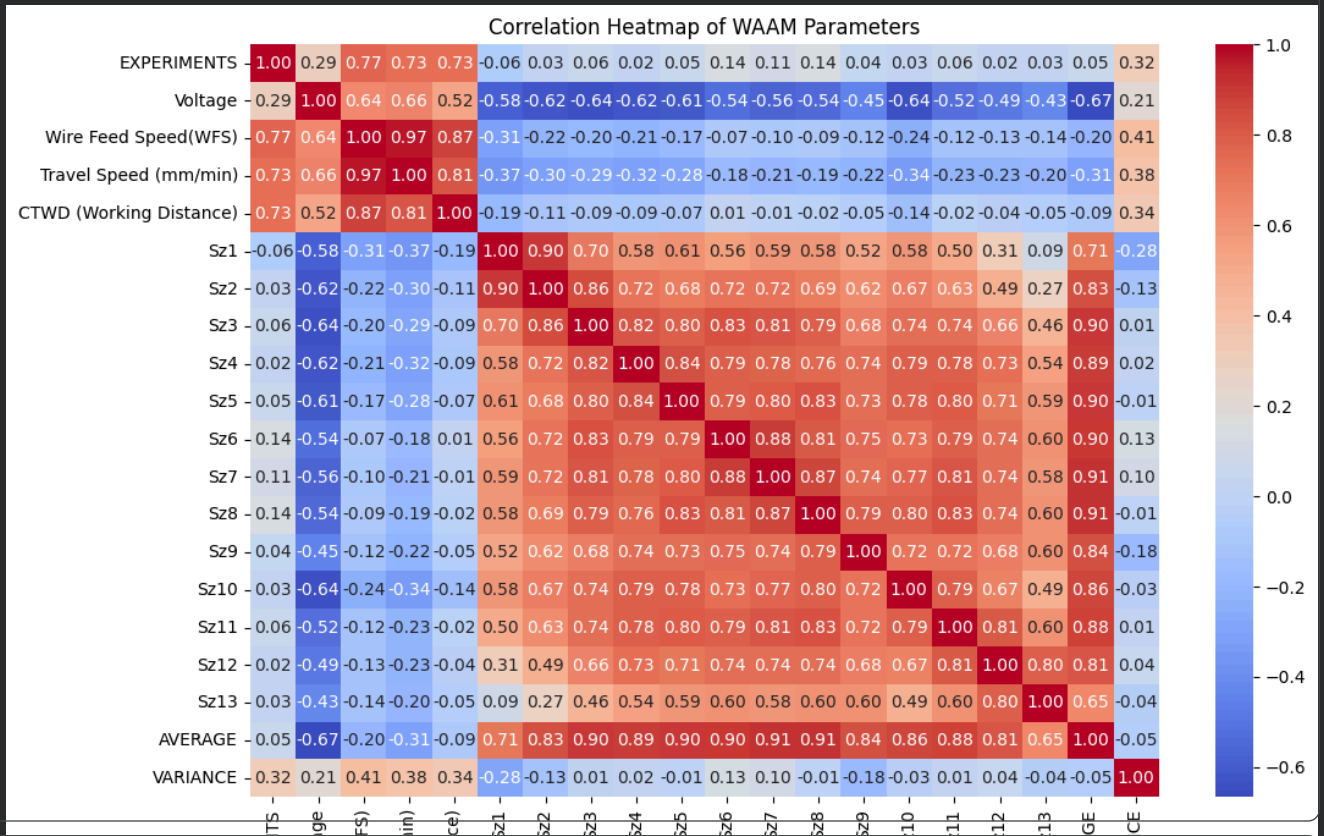
plt.subplot(1,2,2)
sns.histplot(df['VARIANCE'], kde=True, color='red')
plt.title("Distribution of VARIANCE")

plt.tight_layout()
```

```
plt.show()
```

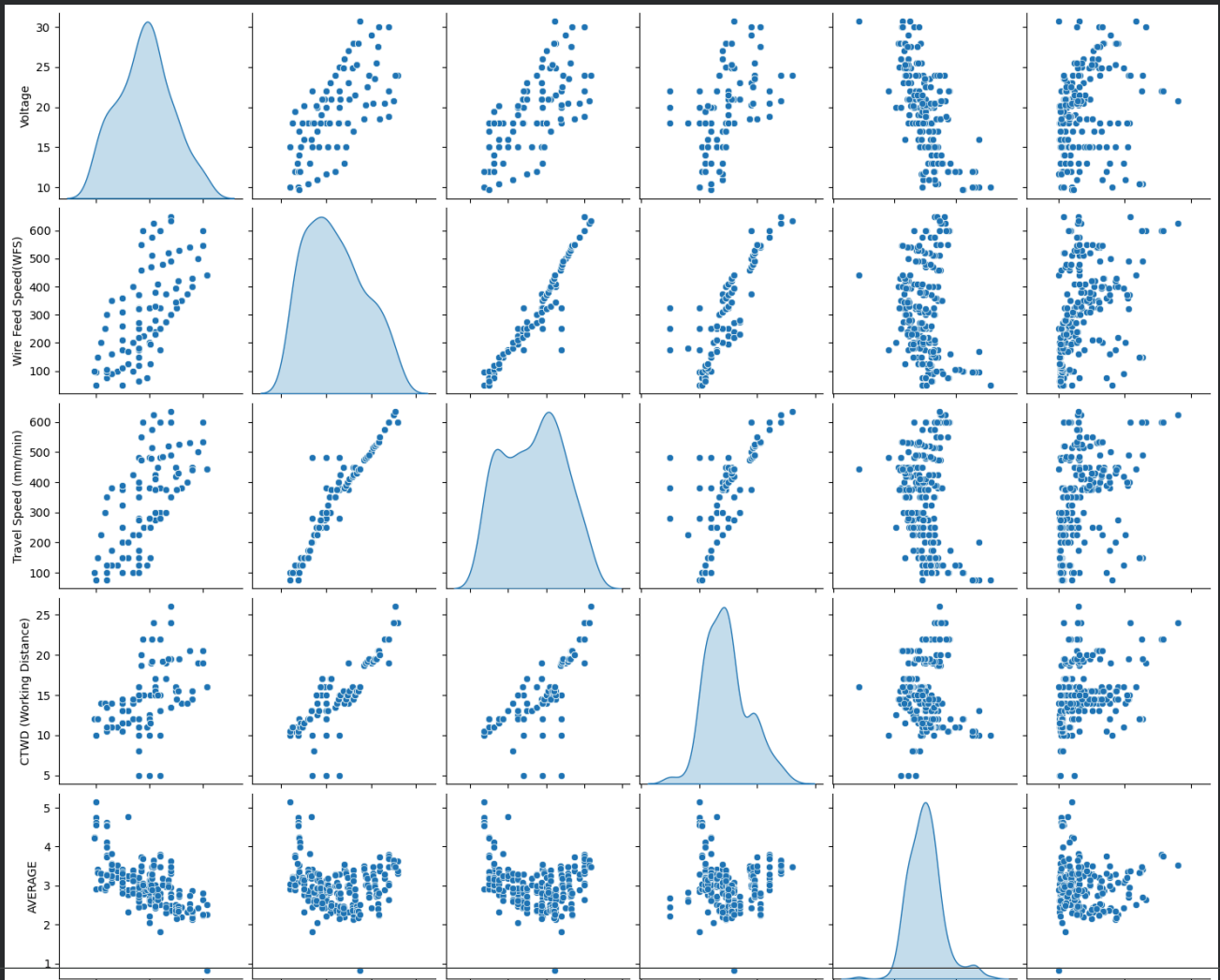


```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap of WAAM Parameters")
plt.show()
```



```
sns.pairplot(df[['Voltage','Wire Feed Speed(WFS)','Travel Speed (mm/min)',
                 'CTWD (Working Distance)','AVERAGE','VARIANCE']],
             diag_kind='kde')
plt.show()
```

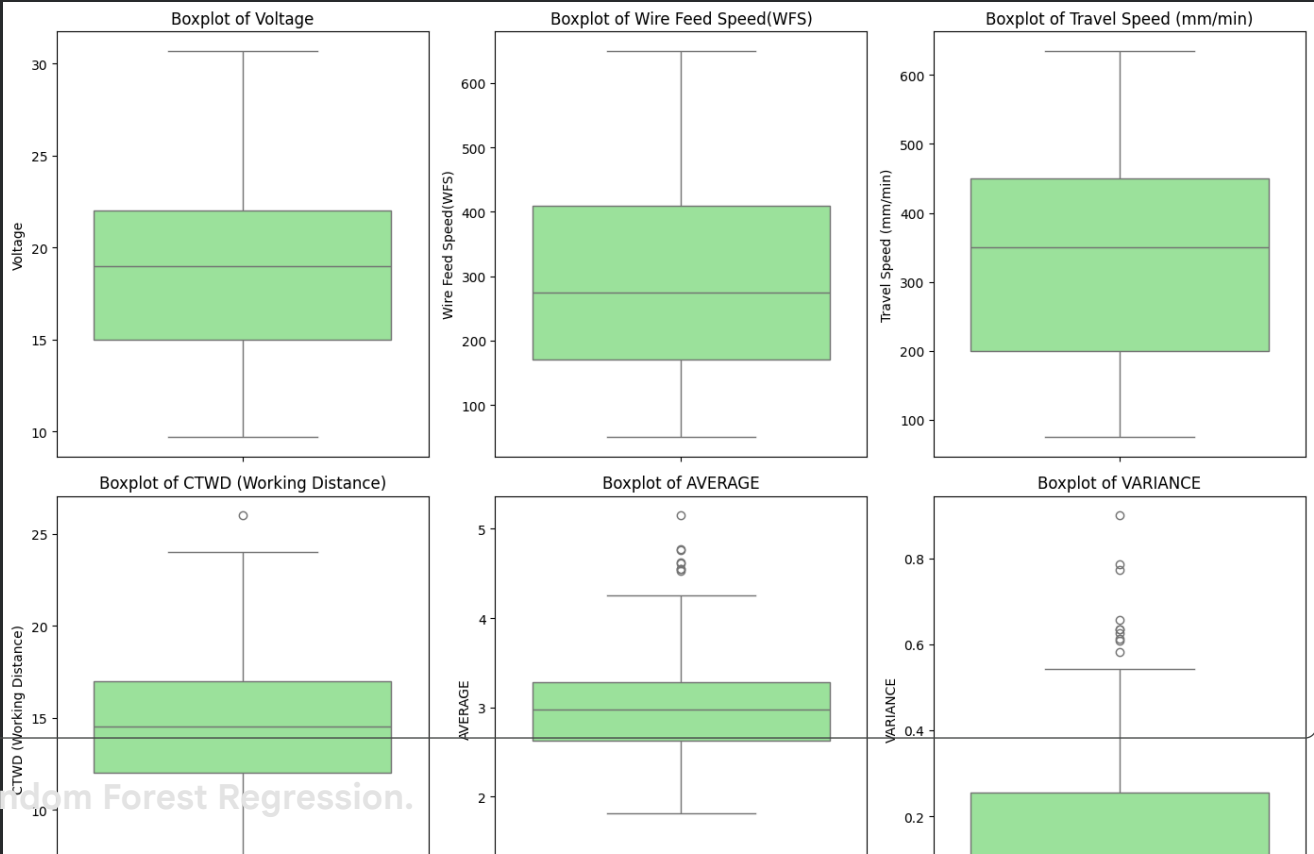
W
Tr
CTW



```
plt.figure(figsize=(14,10))
cols = ['Voltage','Wire Feed Speed(WFS)','Travel Speed (mm/min)',
        'CTWD (Working Distance)','AVERAGE','VARIANCE']

for i, col in enumerate(cols):
    plt.subplot(2,3,i+1)
    plt.boxplot(y=df[col], color='lightgreen')
    plt.title(f"Boxplot of {col}")

plt.tight_layout()
plt.show()
```



Random Forest Regression.

```
# Select Features and Targets
# Inputs = First 4 features
X = df[['Voltage', 'Wire Feed Speed(WFS)', 'Travel Speed (mm/min)', 'CTWD (Working Distance)']]

# Outputs = Average & Variance
y = df[['AVERAGE', 'VARIANCE']]

print("Input shape:", X.shape)
print("Output shape:", y.shape)
```

```
Input shape: (229, 4)
Output shape: (229, 2)
```

```
from sklearn.model_selection import train_test_split
# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
from sklearn.preprocessing import StandardScaler
# Scale Inputs
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
from sklearn.ensemble import RandomForestRegressor
# Training Random Forest Regressor
rf_avg = RandomForestRegressor(n_estimators=300, random_state=42)
rf_var = RandomForestRegressor(n_estimators=300, random_state=42)

# Train models
rf_avg.fit(X_train_scaled, y_train['AVERAGE'])
rf_var.fit(X_train_scaled, y_train['VARIANCE'])
```

```
RandomForestRegressor(n_estimators=300, random_state=42)
```

```
#prediction
y_pred_avg = rf_avg.predict(X_test_scaled)
y_pred_var = rf_var.predict(X_test_scaled)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np
# Evaluation Metrics
# AVERAGE
rmse_avg = np.sqrt(mean_squared_error(y_test['AVERAGE'], y_pred_avg))
mae_avg = mean_absolute_error(y_test['AVERAGE'], y_pred_avg)
r2_avg = r2_score(y_test['AVERAGE'], y_pred_avg)

# VARIANCE
rmse_var = np.sqrt(mean_squared_error(y_test['VARIANCE'], y_pred_var))
mae_var = mean_absolute_error(y_test['VARIANCE'], y_pred_var)
r2_var = r2_score(y_test['VARIANCE'], y_pred_var)

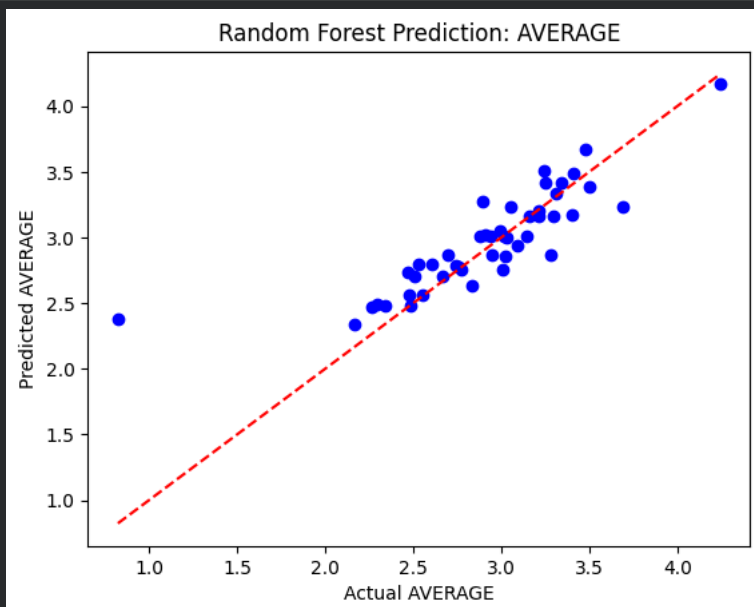
print("==== AVERAGE (Bead Height) ====")
print("RMSE:", rmse_avg)
print("MAE:", mae_avg)
print("R² Score:", r2_avg)

print("\n==== VARIANCE (Bead Stability) ====")
print("RMSE:", rmse_var)
print("MAE:", mae_var)
print("R² Score:", r2_var)
```

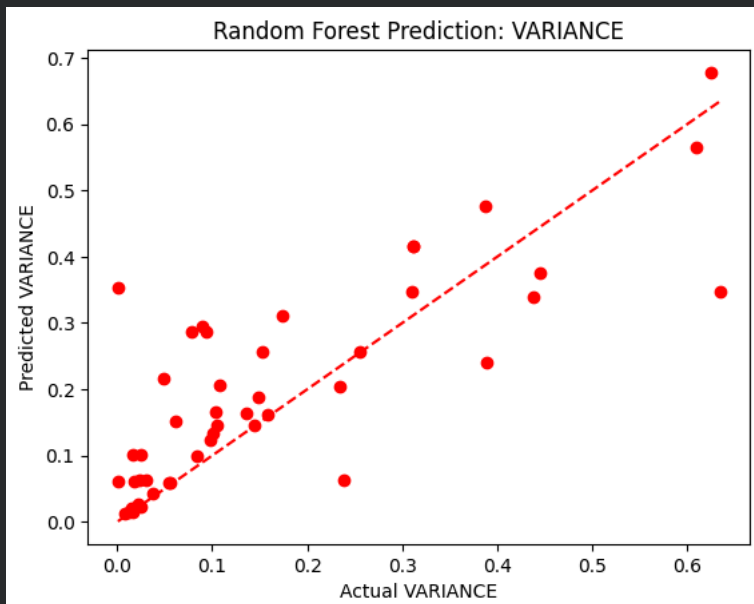
```
==== AVERAGE (Bead Height) ====
RMSE: 0.2880085567811965
MAE: 0.16956072036961944
R² Score: 0.6851073115314978

==== VARIANCE (Bead Stability) ====
RMSE: 0.10832201589862321
MAE: 0.07432176669992319
R² Score: 0.6034158196013664
```

```
# Plot Actual vs Predicted
plt.scatter(y_test['AVERAGE'], y_pred_avg, color='blue')
plt.xlabel("Actual AVERAGE")
plt.ylabel("Predicted AVERAGE")
plt.title("Random Forest Prediction: AVERAGE")
plt.plot([y_test['AVERAGE'].min(), y_test['AVERAGE'].max()],
         [y_test['AVERAGE'].min(), y_test['AVERAGE'].max()],
         'r--')
plt.show()
```



```
# VARIANCE
plt.scatter(y_test['VARIANCE'], y_pred_var, color='red')
plt.xlabel("Actual VARIANCE")
plt.ylabel("Predicted VARIANCE")
plt.title("Random Forest Prediction: VARIANCE")
plt.plot([y_test['VARIANCE'].min(), y_test['VARIANCE'].max()],
         [y_test['VARIANCE'].min(), y_test['VARIANCE'].max()],
         'r--')
plt.show()
```



▼ Deep Learning Model Using Keras (TensorFlow)

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
```

```
# Building the Neural Network Model
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dense(32, activation='relu'),
```

```

Dense(16, activation='relu'),
Dense(2) # output layer: [AVERAGE, VARIANCE]
])

```

```

model.compile(optimizer='adam', loss='mse')
model.summary()

```

```

/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim`
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"

```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	320
dense_1 (Dense)	(None, 32)	2,080
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 2)	34

```

Total params: 2,962 (11.57 KB)
Trainable params: 2,962 (11.57 KB)
Non-trainable params: 0 (0.00 B)

```

```

# Training the Model
early_stop = EarlyStopping(monitor='val_loss', patience=20, restore_best_weights=True)

history = model.fit(
    X_train_scaled, y_train,
    validation_split=0.2,
    epochs=300,
    batch_size=16,
    callbacks=[early_stop],
    verbose=1
)

```

```

Epoch 1/300
10/10 ————— 2s 25ms/step - loss: 4.8603 - val_loss: 4.4735
Epoch 2/300
10/10 ————— 0s 10ms/step - loss: 4.1175 - val_loss: 4.0124
Epoch 3/300
10/10 ————— 0s 9ms/step - loss: 3.5820 - val_loss: 3.3194
Epoch 4/300
10/10 ————— 0s 9ms/step - loss: 3.0604 - val_loss: 2.4653
Epoch 5/300
10/10 ————— 0s 9ms/step - loss: 2.1510 - val_loss: 1.5394
Epoch 6/300
10/10 ————— 0s 9ms/step - loss: 1.2125 - val_loss: 0.7409
Epoch 7/300
10/10 ————— 0s 9ms/step - loss: 0.6547 - val_loss: 0.4444
Epoch 8/300
10/10 ————— 0s 9ms/step - loss: 0.4911 - val_loss: 0.4212
Epoch 9/300
10/10 ————— 0s 9ms/step - loss: 0.5156 - val_loss: 0.3685
Epoch 10/300
10/10 ————— 0s 9ms/step - loss: 0.3891 - val_loss: 0.3227
Epoch 11/300
10/10 ————— 0s 9ms/step - loss: 0.3602 - val_loss: 0.2912
Epoch 12/300
10/10 ————— 0s 9ms/step - loss: 0.2828 - val_loss: 0.2632
Epoch 13/300
10/10 ————— 0s 9ms/step - loss: 0.2690 - val_loss: 0.2362
Epoch 14/300
10/10 ————— 0s 9ms/step - loss: 0.2889 - val_loss: 0.2128
Epoch 15/300
10/10 ————— 0s 14ms/step - loss: 0.2142 - val_loss: 0.1938
Epoch 16/300
10/10 ————— 0s 14ms/step - loss: 0.2087 - val_loss: 0.1831
Epoch 17/300
10/10 ————— 0s 14ms/step - loss: 0.1877 - val_loss: 0.1627
Epoch 18/300
10/10 ————— 0s 16ms/step - loss: 0.1658 - val_loss: 0.1500
Epoch 19/300
10/10 ————— 0s 13ms/step - loss: 0.1328 - val_loss: 0.1405
Epoch 20/300
10/10 ————— 0s 16ms/step - loss: 0.1298 - val_loss: 0.1323
Epoch 21/300
10/10 ————— 0s 12ms/step - loss: 0.1249 - val_loss: 0.1268

```

```

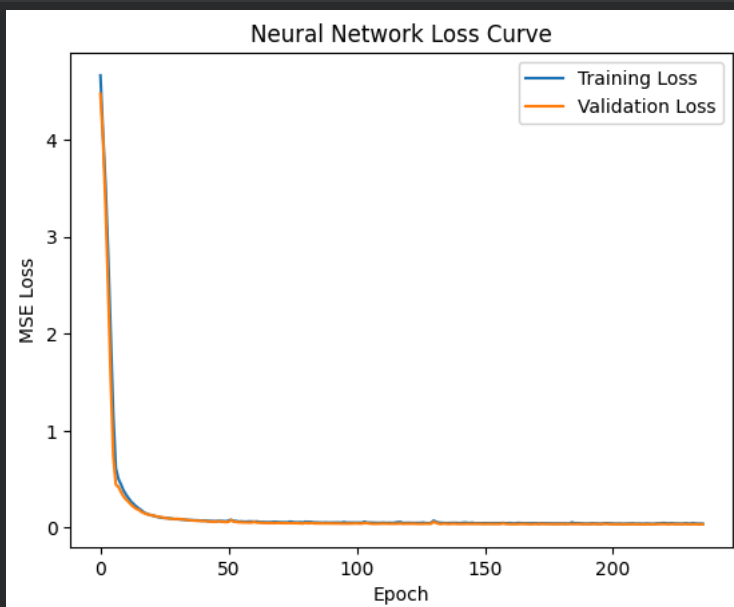
Epoch 22/300
10/10 — 0s 16ms/step - loss: 0.1042 - val_loss: 0.1235
Epoch 23/300
10/10 — 0s 13ms/step - loss: 0.1179 - val_loss: 0.1159
Epoch 24/300
10/10 — 0s 17ms/step - loss: 0.0949 - val_loss: 0.1113
Epoch 25/300
10/10 — 0s 13ms/step - loss: 0.1007 - val_loss: 0.1070
Epoch 26/300
10/10 — 0s 18ms/step - loss: 0.0965 - val_loss: 0.1024
Epoch 27/300
10/10 — 0s 15ms/step - loss: 0.0953 - val_loss: 0.0978
Epoch 28/300
10/10 — 0s 11ms/step - loss: 0.0748 - val_loss: 0.0943
Epoch 29/300

```

```

# Plot Training & Validation Loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title("Neural Network Loss Curve")
plt.xlabel("Epoch")
plt.ylabel("MSE Loss")
plt.legend()
plt.show()

```



```

# Predictions
y_pred_d1 = model.predict(X_test_scaled)

y_pred_avg_d1 = y_pred_d1[:, 0]
y_pred_var_d1 = y_pred_d1[:, 1]

```

```

2/2 — 0s 57ms/step

```

```

# Evaluation Metrics
# AVERAGE
rmse_avg_d1 = np.sqrt(mean_squared_error(y_test['AVERAGE'], y_pred_avg_d1))
mae_avg_d1 = mean_absolute_error(y_test['AVERAGE'], y_pred_avg_d1)
r2_avg_d1 = r2_score(y_test['AVERAGE'], y_pred_avg_d1)

# VARIANCE
rmse_var_d1 = np.sqrt(mean_squared_error(y_test['VARIANCE'], y_pred_var_d1))
mae_var_d1 = mean_absolute_error(y_test['VARIANCE'], y_pred_var_d1)
r2_var_d1 = r2_score(y_test['VARIANCE'], y_pred_var_d1)

print("==== DEEP LEARNING: AVERAGE ====")
print("RMSE:", rmse_avg_d1)
print("MAE:", mae_avg_d1)
print("R²:", r2_avg_d1)

print("\n==== DEEP LEARNING: VARIANCE ====")

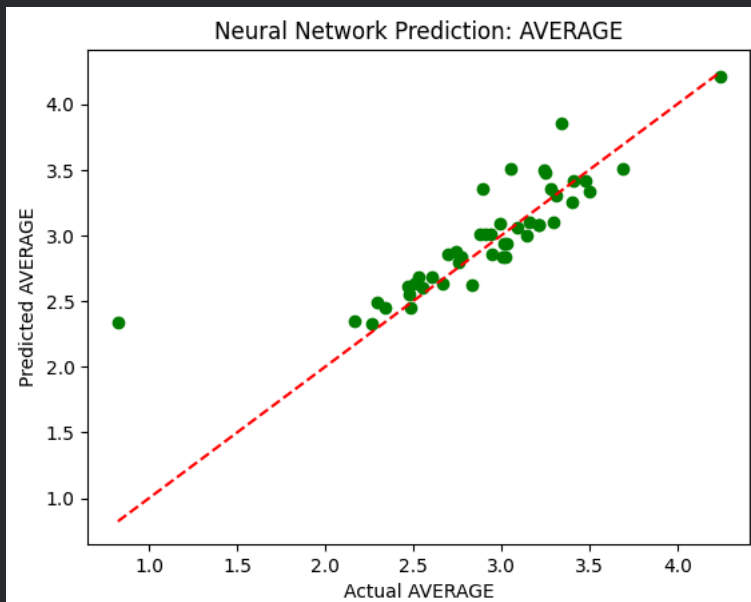
```

```
print("RMSE:", rmse_var_dl)
print("MAE:", mae_var_dl)
print("R²:", r2_var_dl)
```

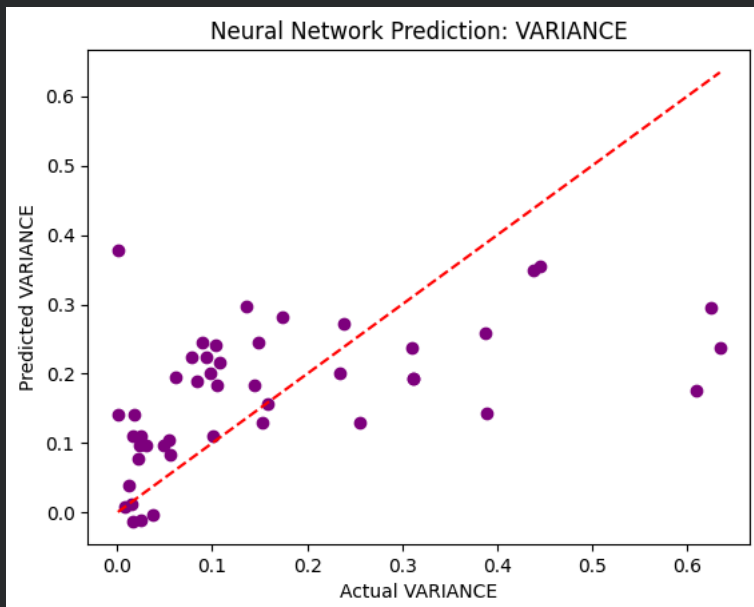
```
===== DEEP LEARNING: AVERAGE =====
RMSE: 0.28181943760305034
MAE: 0.16608528325188368
R²: 0.6984955802915215
```

```
===== DEEP LEARNING: VARIANCE =====
RMSE: 0.1475503371602017
MAE: 0.10935995907968381
R²: 0.2641617975215188
```

```
#scatter plot
plt.scatter(y_test['AVERAGE'], y_pred_avg_dl, color='green')
plt.plot([y_test['AVERAGE'].min(), y_test['AVERAGE'].max()],
         [y_test['AVERAGE'].min(), y_test['AVERAGE'].max()],
         'r--')
plt.xlabel("Actual AVERAGE")
plt.ylabel("Predicted AVERAGE")
plt.title("Neural Network Prediction: AVERAGE")
plt.show()
```



```
# Variance
plt.scatter(y_test['VARIANCE'], y_pred_var_dl, color='purple')
plt.plot([y_test['VARIANCE'].min(), y_test['VARIANCE'].max()],
         [y_test['VARIANCE'].min(), y_test['VARIANCE'].max()],
         'r--')
plt.xlabel("Actual VARIANCE")
plt.ylabel("Predicted VARIANCE")
plt.title("Neural Network Prediction: VARIANCE")
plt.show()
```



Comparison

```
# Comparison Table
comparison = pd.DataFrame({
    "Model": ["Random Forest (AVG)", "Neural Network (AVG)",
              "Random Forest (VAR)", "Neural Network (VAR)"],
    "RMSE": [rmse_avg, rmse_avg_d1, rmse_var, rmse_var_d1],
    "MAE": [mae_avg, mae_avg_d1, mae_var, mae_var_d1],
    "R² Score": [r2_avg, r2_avg_d1, r2_var, r2_var_d1]
})
```

```
comparison
```

	Model	RMSE	MAE	R² Score
0	Random Forest (AVG)	0.288009	0.169561	0.685107
1	Neural Network (AVG)	0.281819	0.166085	0.698496
2	Random Forest (VAR)	0.108322	0.074322	0.603416
3	Neural Network (VAR)	0.147550	0.109360	0.264162

```
# Feature Importance for Random Forest
importances_avg = rf_avg.feature_importances_
importances_var = rf_var.feature_importances_

features = ['Voltage', 'Wire Feed Speed', 'Travel Speed', 'CTWD']

for name, importance in zip(features, importances_avg):
    print(f"AVG - {name}: {importance:.4f}")

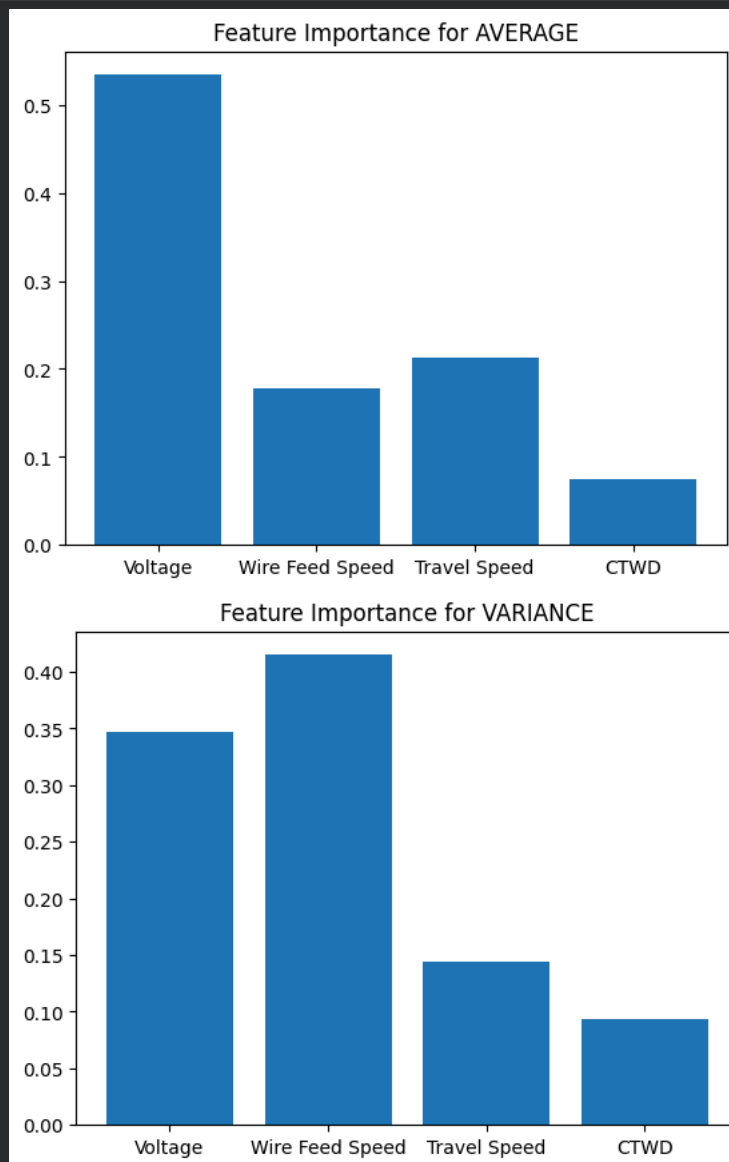
for name, importance in zip(features, importances_var):
    print(f"VAR - {name}: {importance:.4f}")
```

```
AVG - Voltage: 0.5339
AVG - Wire Feed Speed: 0.1778
AVG - Travel Speed: 0.2135
AVG - CTWD: 0.0748
VAR - Voltage: 0.3469
VAR - Wire Feed Speed: 0.4151
VAR - Travel Speed: 0.1439
VAR - CTWD: 0.0941
```

```
# Bar plot
plt.bar(features, importances_avg)
plt.title("Feature Importance for AVERAGE")
```

```
plt.show()

plt.bar(features, importances_var)
plt.title("Feature Importance for VARIANCE")
plt.show()
```



```
import joblib

joblib.dump(rf_avg, "rf_avg_model.pkl")
joblib.dump(rf_var, "rf_var_model.pkl")
```

```
['rf_var_model.pkl']
```

```
model.save("waam_nn_model.h5")
model.save("waam_nn_model.keras")
```