Prediction and optimisation of Nozzle diameter , Nozzle temperature , Print speed , Stress level in additive-manufactured 3D-printed Polylactic acid biomaterials

Final Year Project May-2025

Aashish zeruba s

B.Tech Mechanical

Table of contents:

INTRODUCTION:

Additive Manufacturing:

- Additive manufacturing is the process of creating an object by building it one layer at a time. It is the opposite of subtractive manufacturing, in which an object is created by cutting away at a solid block of material until the final product is complete.
- Additive manufacturing helps in addressing the challenges in conventional manufacturin. processes.

PLA Material (Polylactic acid):

- Polylactic acid, also known as PLA, is a thermoplastic monomer derived from renewable, organic sources such as corn starch or sugar cane.

Fatigue Life:

- Fatigue life is the number of cycles a material can withstand before failing under cyclic loading.

Machine Learning:

- The Machine Learning (ML) can train and test the available data to predict and optimize the data through various techniques such as regression analysis.

Problem Statement:

- The Machine Learning (ML) can train and test the available data to predict and optimize the data through various techniques such as regression analysis.
- Additive manufacturing helps in addressing the challenges in conventional manufacturing processes.
- This project aims to predict and optimize Nozzle diameter , Nozzle temperature , Print speed , Stress level in additive-manufactured 3D-printed Polylactic acid biomaterials.

Objective:

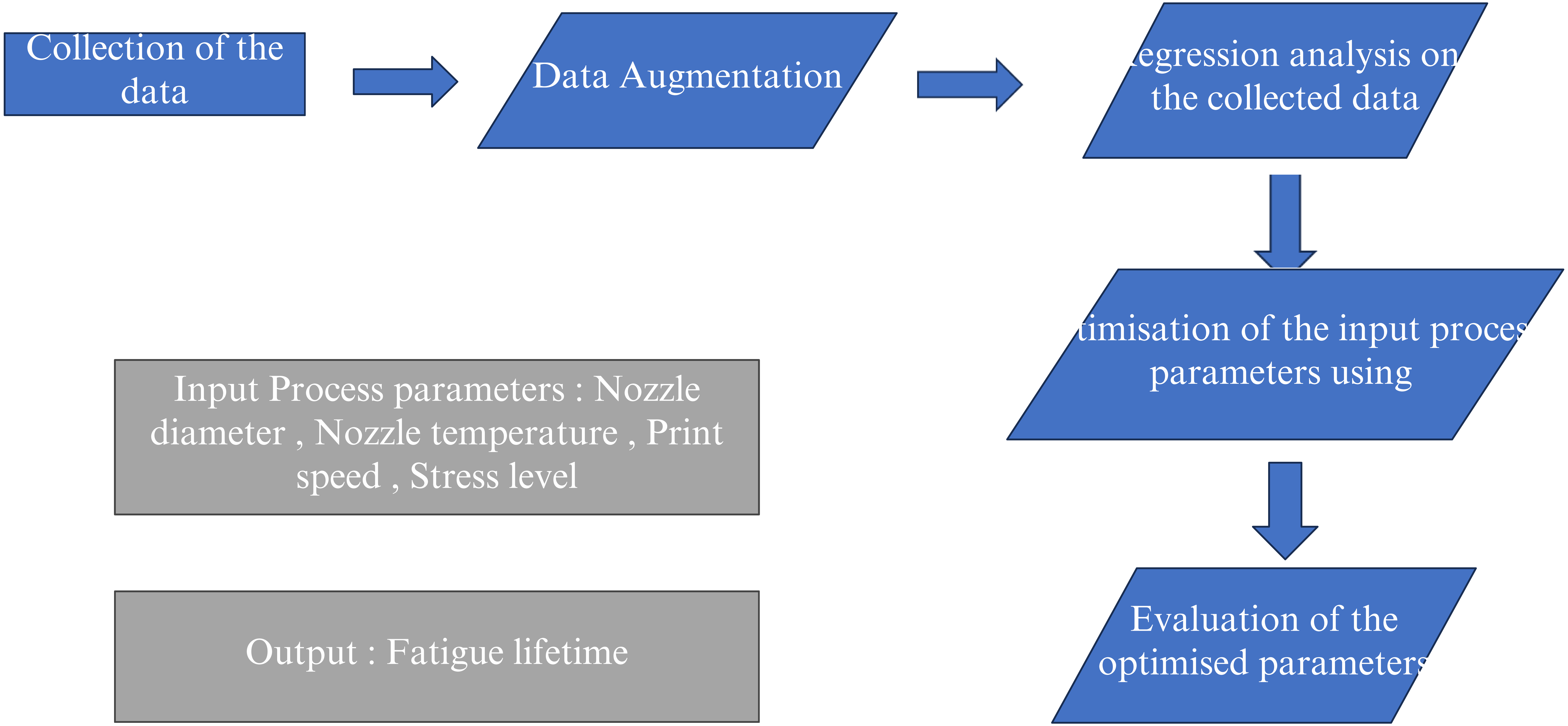
- Understanding the concept of Machine learning
- Collect the data set
- Create codes to optimize using the regression approach
- Optimization of the input process parameters
- Evaluation of the predicted result

Literature Survey:

S. No	Title	Inference
1	biomaterials under fully-reversed rotating-bending	been presented for the additive-manufactured 3D-printed Poly lactic acid (PLA) biomaterials under fully-
2	Fatigue Properties of Additively Manufactured Ti-6Al-4V	identify the behavior of the relationship between the input and output parameters, the model is developed on both linear multi-regression analysis and non linear
3	additive manufacturing process for enhancing mechanical properties	surface approach, and Taguchi method. When the optimal solutions of each technique were studied, the

4	Single and Multi-Objective Optimisation of Processing Parameters for Fused Deposition Modelling in 3D Printing Technology V.H. Nguyen, T.N. Huynh, T.P. Nguyen and T.T. Tran (2020)	This paper presents practice and application of design of experiment techniques and genetic algorithm in single and multi-objective optimization with low cost, robustness, and high effectiveness through 3D printing case studies. 3D printing brings many benefits for engineering design, product development, and production process.
5	A Parametric study of 3D printed polymer gears Ye Zhang , Ken Mao , Simon Leigh , Guotao Ma (2020)	It deals with the selection of the printing process parameters on the dynamic performance of the components such as spur gears. A genetic algorithm (GA) based artificial neural network (ANN) has been used for this purpose.
6	Modelling and parametric optimization of FDM 3D printing process using hybrid techniques for enhancing dimensional preciseness Sandeep Deswal , Rajan Narang , Deepak Chhabra (2019)	In this study, significant process parameters (layer thickness, build orientation, infill density and number of contours) are optimized for enhancing the magnitude/dimensional preciseness of fused deposition modelling (FDM) devise units. Hybrid statistical tools such as response surface methodology–genetic algorithm (RSM–GA), artificial neural network (ANN) and artificial neural network-genetic algorithm (ANN-GA)

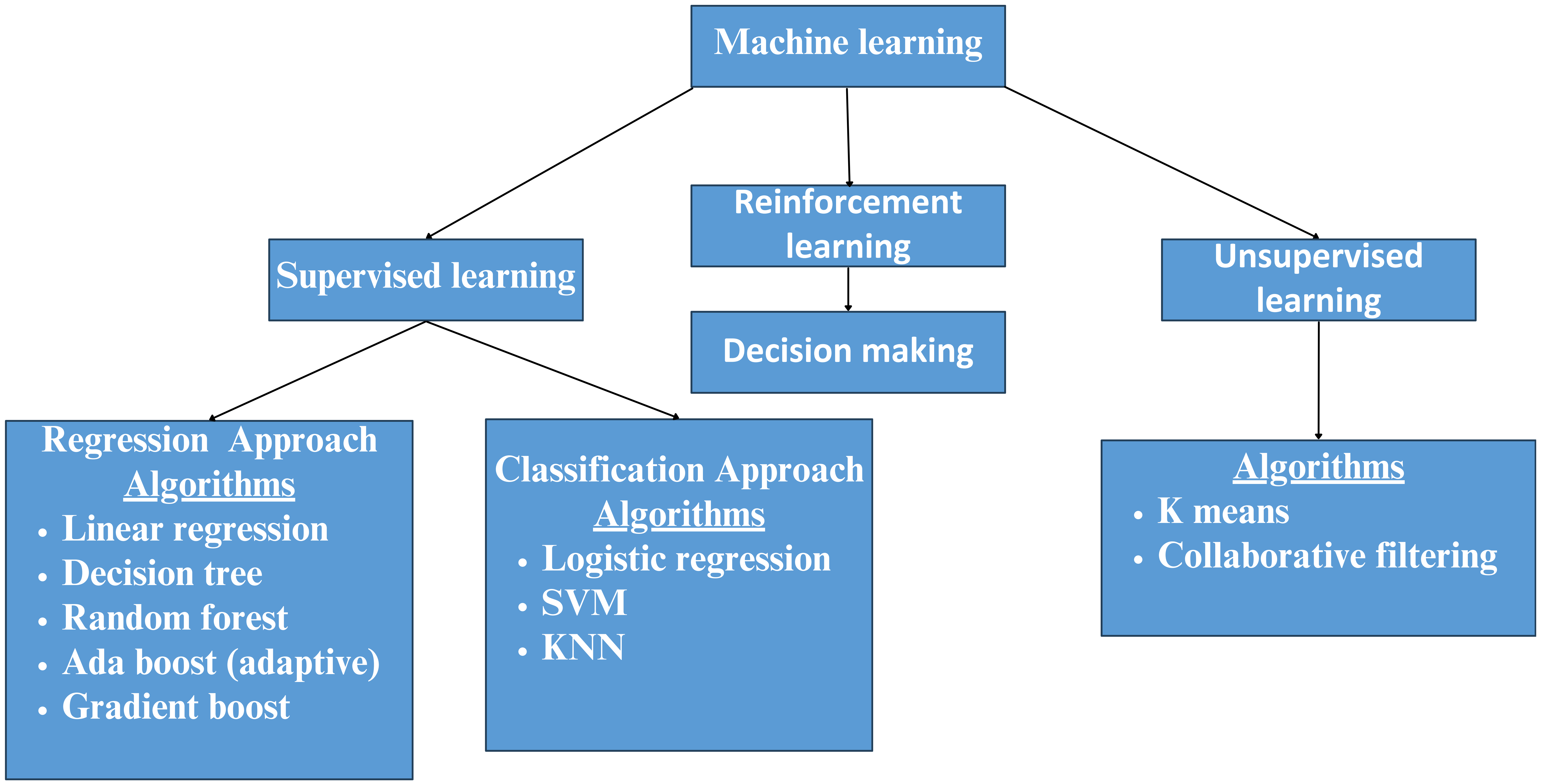
Methodology:


Collection of the data
Data Augmentation
egression analysis on the collected data
timisation of the input proces parameters using
Evaluation of the optimised parameters

Input Process parameters : Nozzle diameter , Nozzle temperature , Print speed , Stress level

Output : Fatigue lifetime

Machine Learning Concept:



Recap >>>

Regression analysis is a statistical method used to examine the relationship between a dependent variable and one or more independent variables. It helps in understanding how the dependent variable changes when one or more independent variables are varied.

Linear regression is a data analysis technique that predicts the value of unknown data by using another related and known data value. It mathematically models the unknown or dependent variable and the known or independent variable as a linear equation.

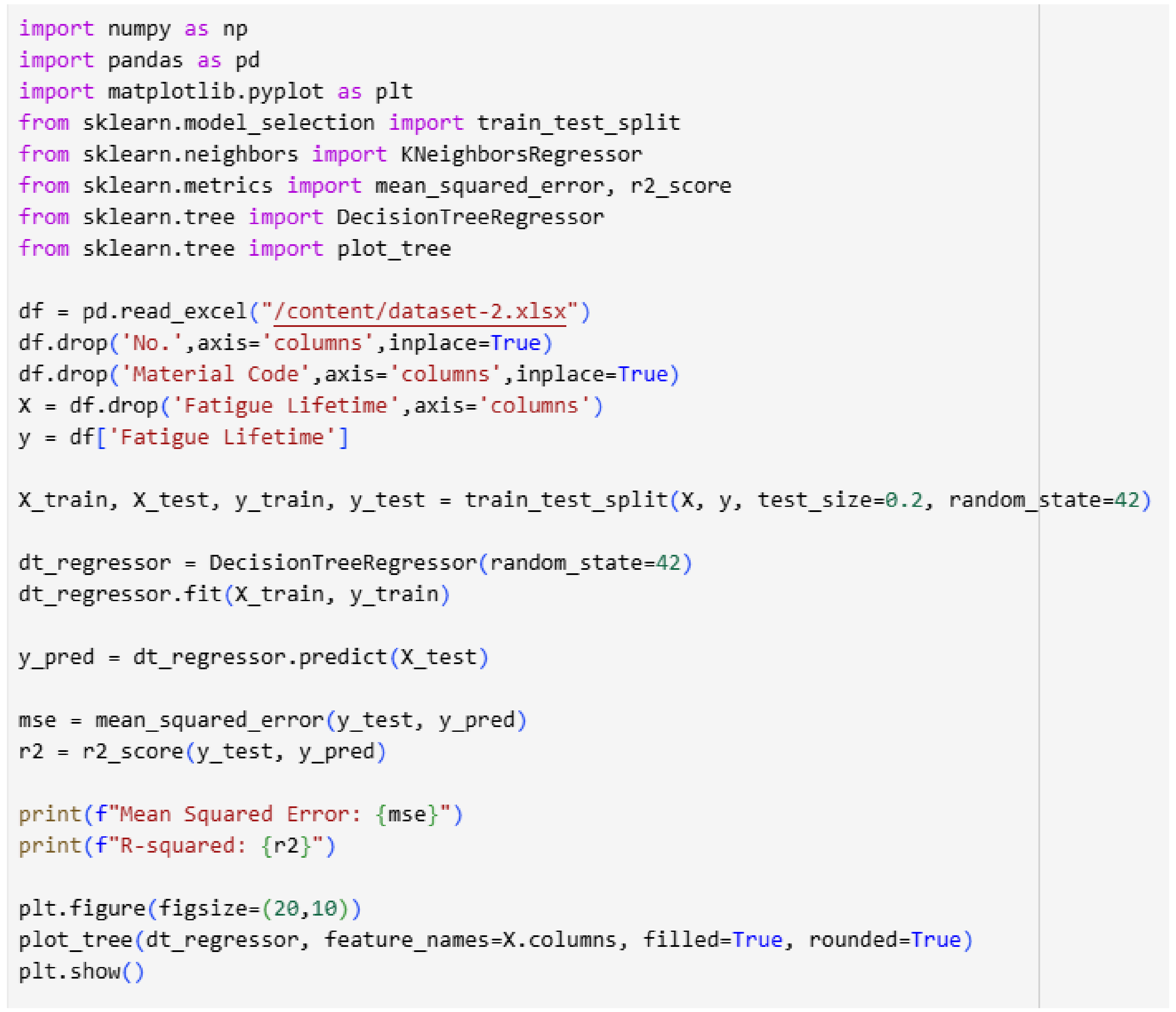
K-Nearest Neighbors (KNN) is a simple, non-parametric machine learning algorithm used for classification and regression tasks. It works by finding the K closest data points (neighbors) to a given query point and making predictions based on these neighbors.

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that best separates data points into different classes.

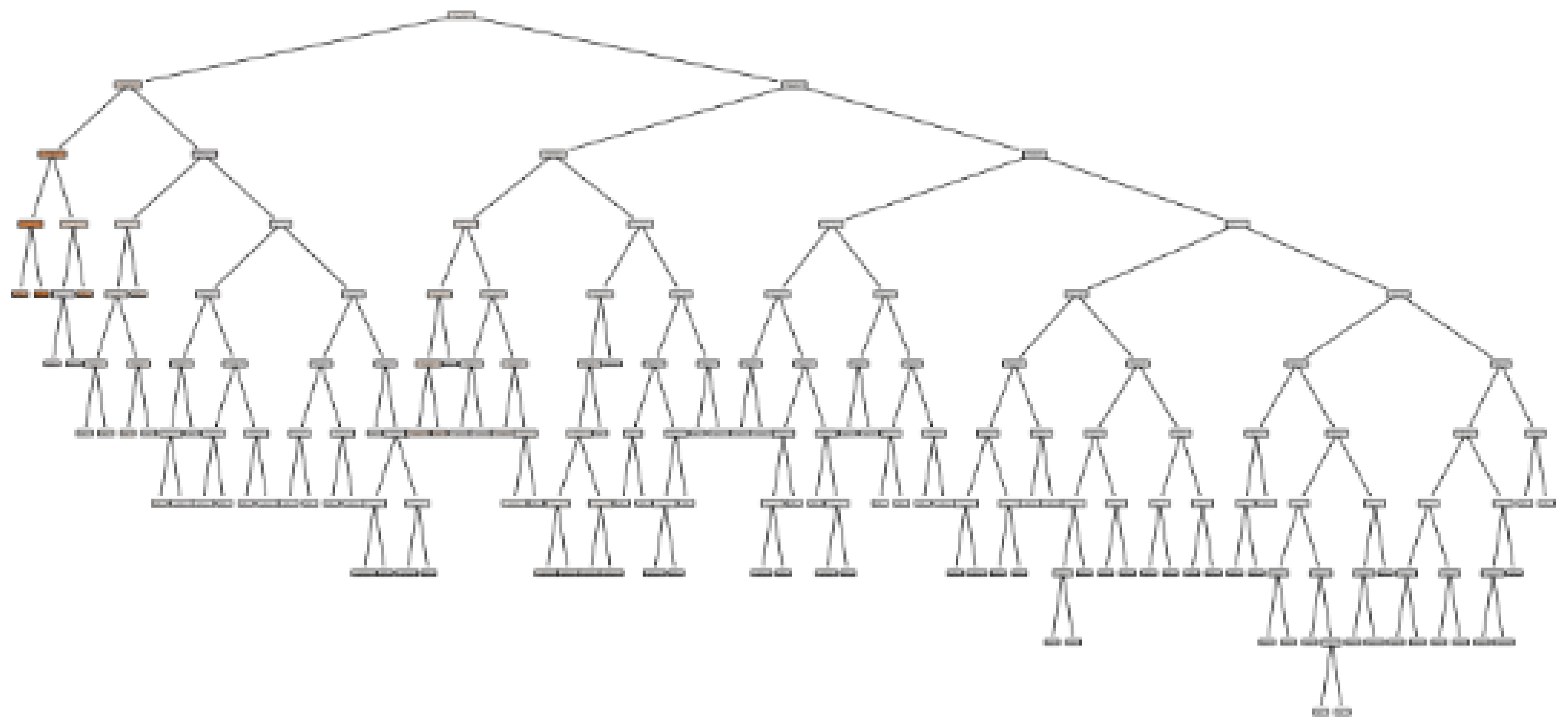
Decision Tree is a supervised machine learning algorithm used for classification and regression tasks. It represents decisions and their possible consequences in a tree-like structure, making it easy to interpret and visualize.

Random Forest is a machine learning algorithm used for classification and regression tasks. It is an ensemble method that builds multiple decision trees and combines their outputs to improve accuracy and reduce overfitting.

Decision Tree regression method:

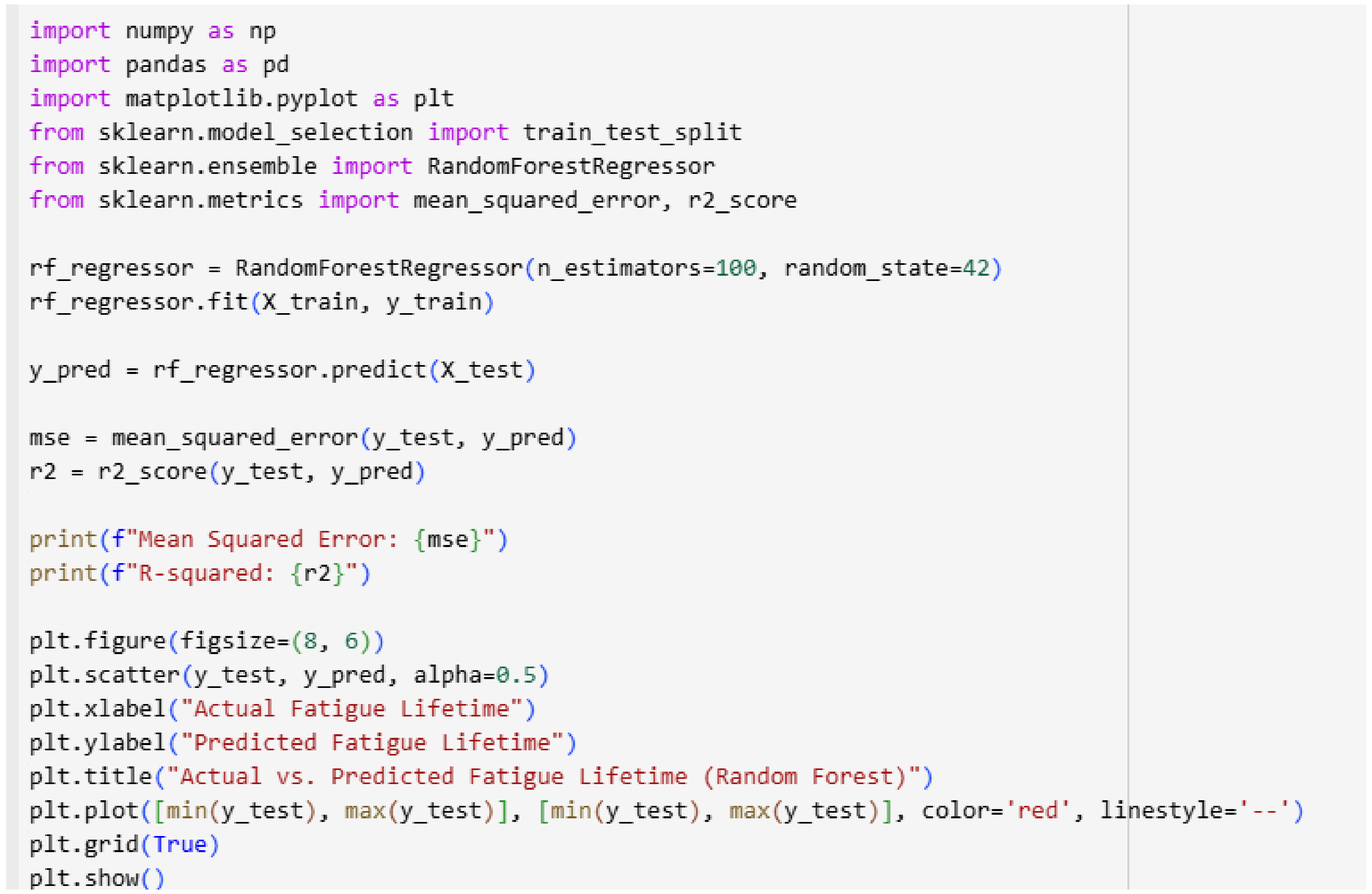


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import plot_tree

df = pd.read_excel("/content/dataset-2.xlsx")
df.drop('No.',axis='columns',inplace=True)
df.drop('Material Code',axis='columns',inplace=True)
X = df.drop('Fatigue Lifetime',axis='columns')
y = df['Fatigue Lifetime']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

dt_regressor = DecisionTreeRegressor(random_state=42)
dt_regressor.fit(X_train, y_train)

y_pred = dt_regressor.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

plt.figure(figsize=(20,10))
plot_tree(dt_regressor, feature_names=X.columns, filled=True, rounded=True)
plt.show()
```

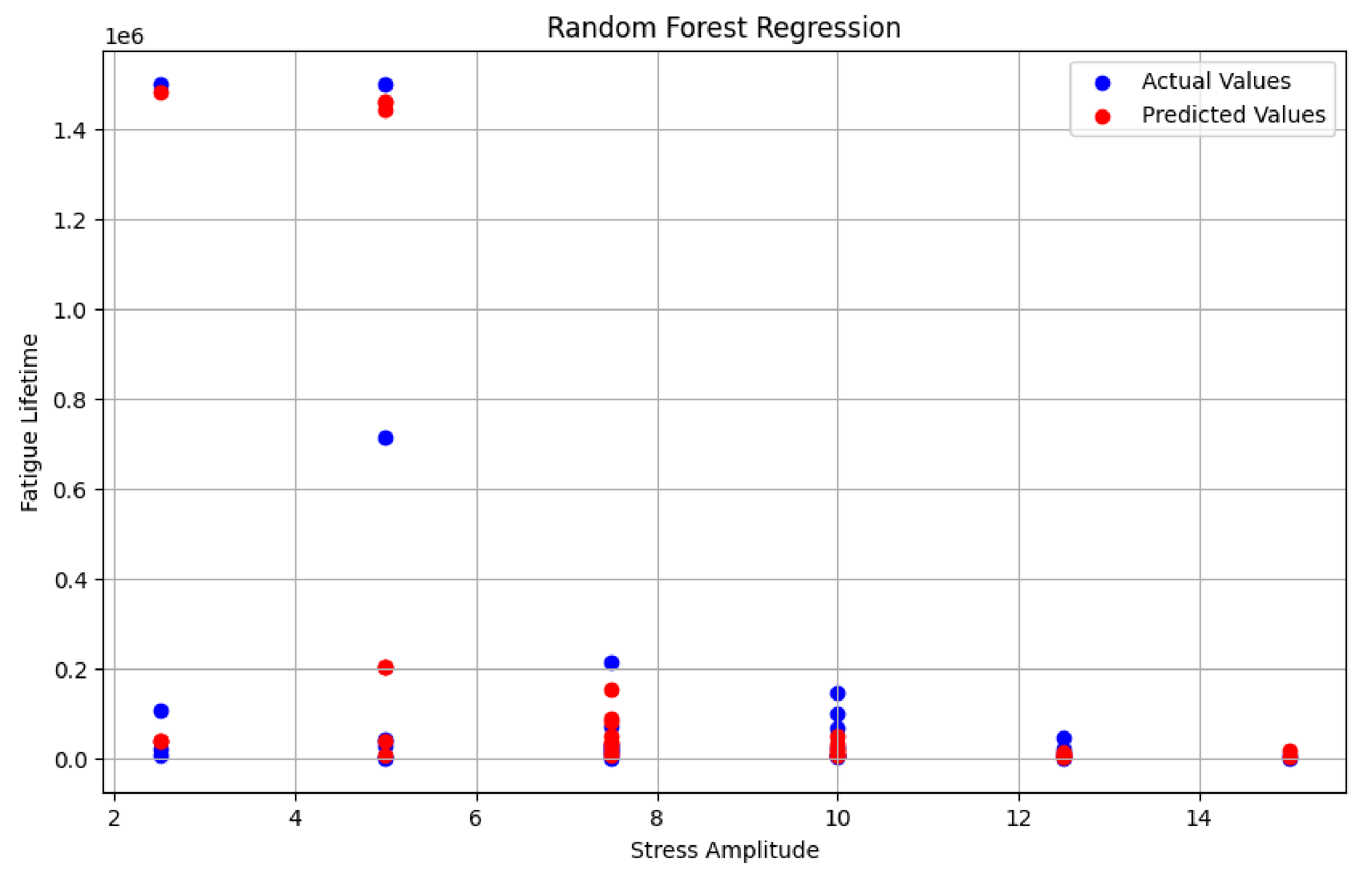


```
Mean Squared Error: 216171860111.1111
R-squared: -0.6959164689929145
```

Random forest regression method:

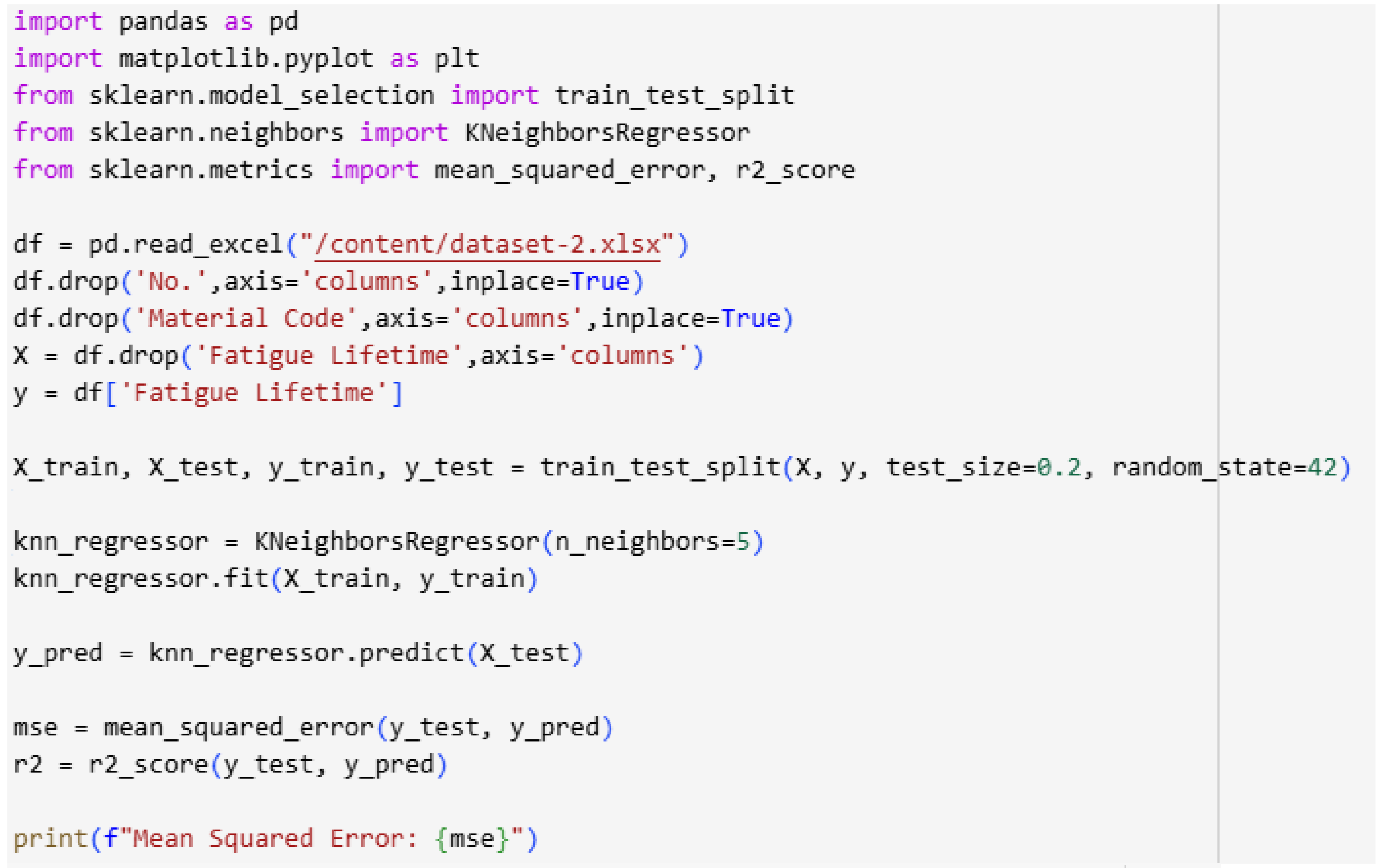


```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)

y_pred = rf_regressor.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel("Actual Fatigue Lifetime")
plt.ylabel("Predicted Fatigue Lifetime")
plt.title("Actual vs. Predicted Fatigue Lifetime (Random Forest)")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.grid(True)
plt.show()
```

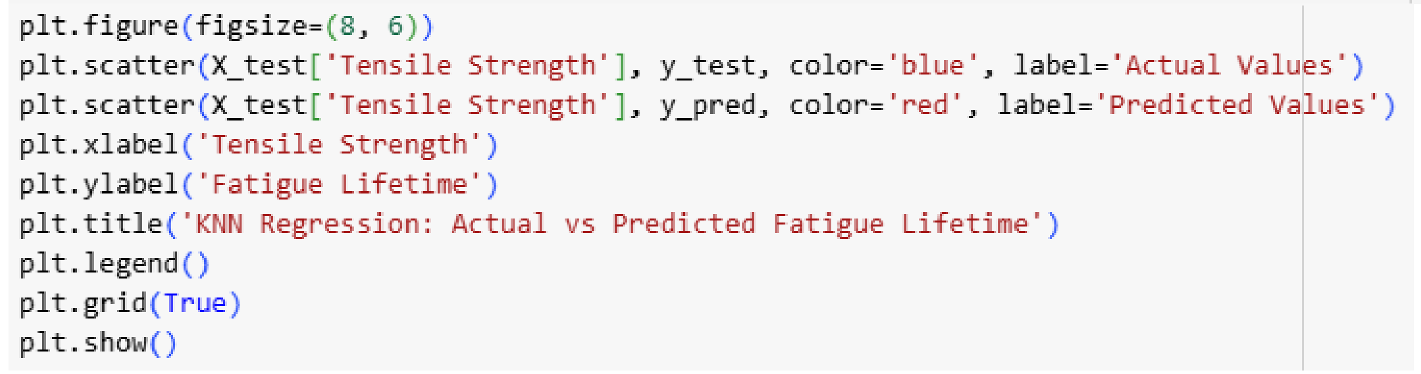


Mean Squared Error: 187791164607.99966
R-squared: -0.47326358123749324

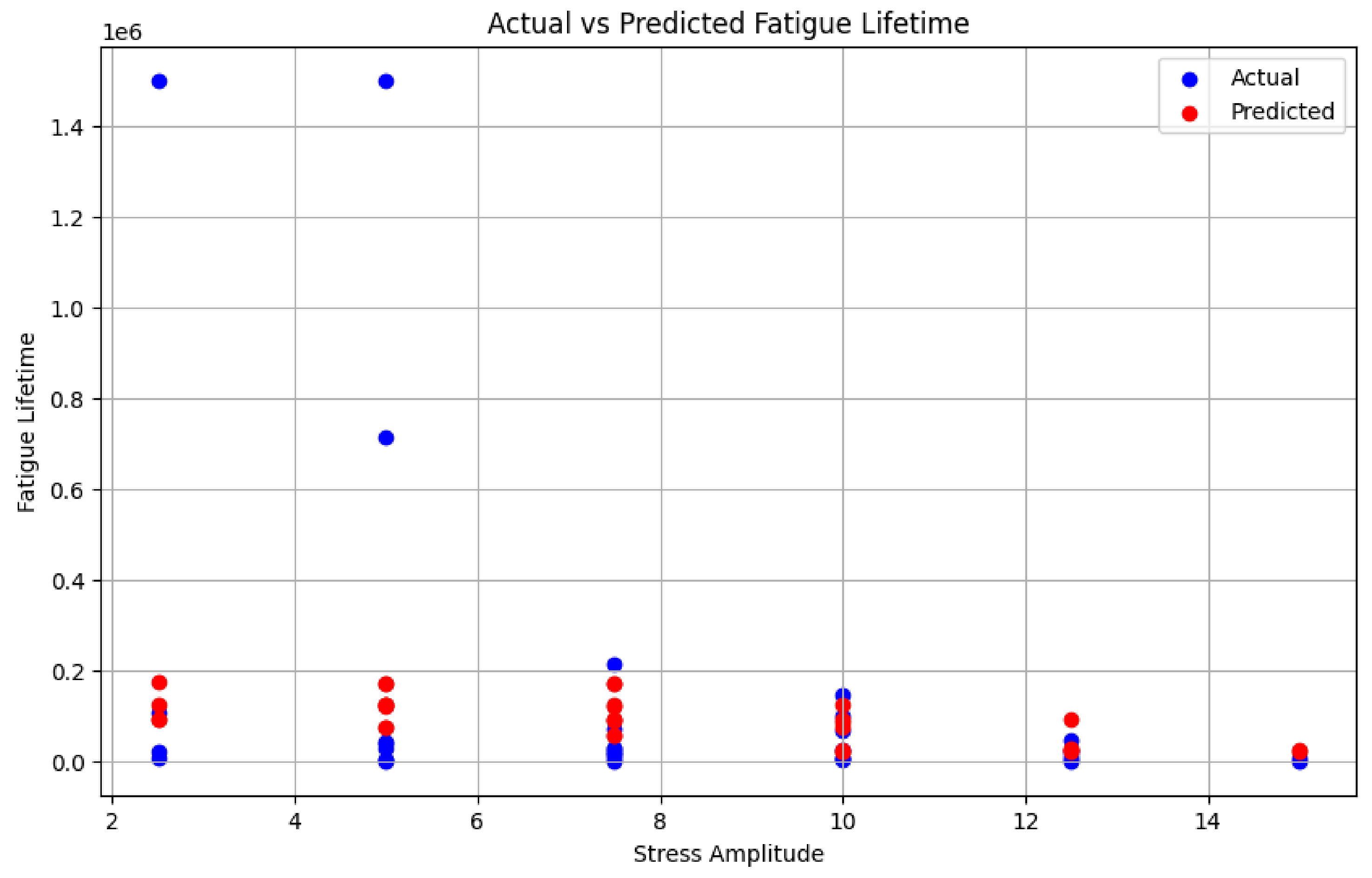
KNN regression method:



```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_excel("/content/dataset-2.xlsx")
df.drop('No.',axis='columns',inplace=True)
df.drop('Material Code',axis='columns',inplace=True)
X = df.drop('Fatigue Lifetime',axis='columns')
y = df['Fatigue Lifetime']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

knn_regressor = KNeighborsRegressor(n_neighbors=5)
knn_regressor.fit(X_train, y_train)

y_pred = knn_regressor.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
```

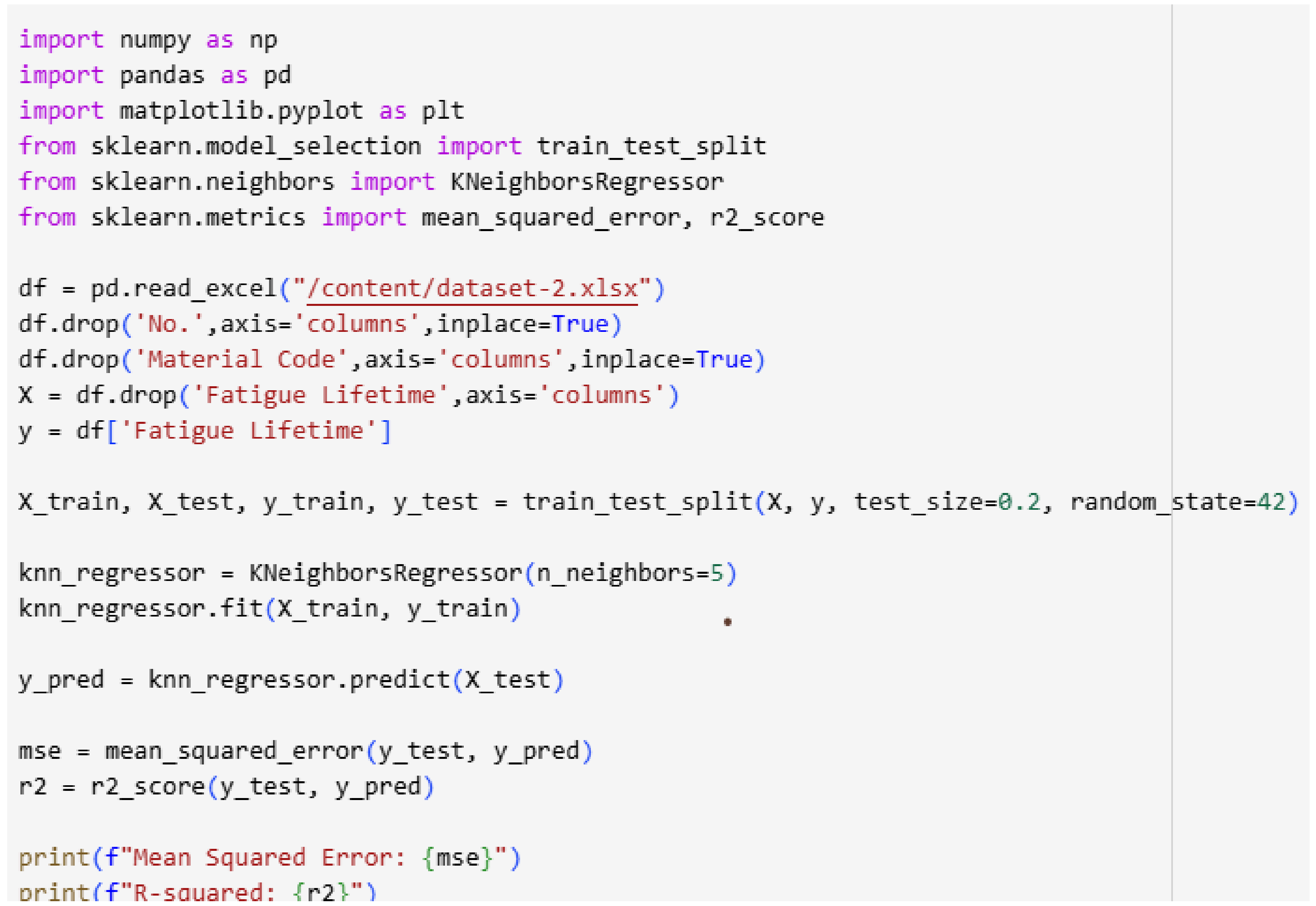


```
plt.figure(figsize=(8, 6))
plt.scatter(X_test['Tensile Strength'], y_test, color='blue', label='Actual Values')
plt.scatter(X_test['Tensile Strength'], y_pred, color='red', label='Predicted Values')
plt.xlabel('Tensile Strength')
plt.ylabel('Fatigue Lifetime')
plt.title('KNN Regression: Actual vs Predicted Fatigue Lifetime')
plt.legend()
plt.grid(True)
plt.show()
```

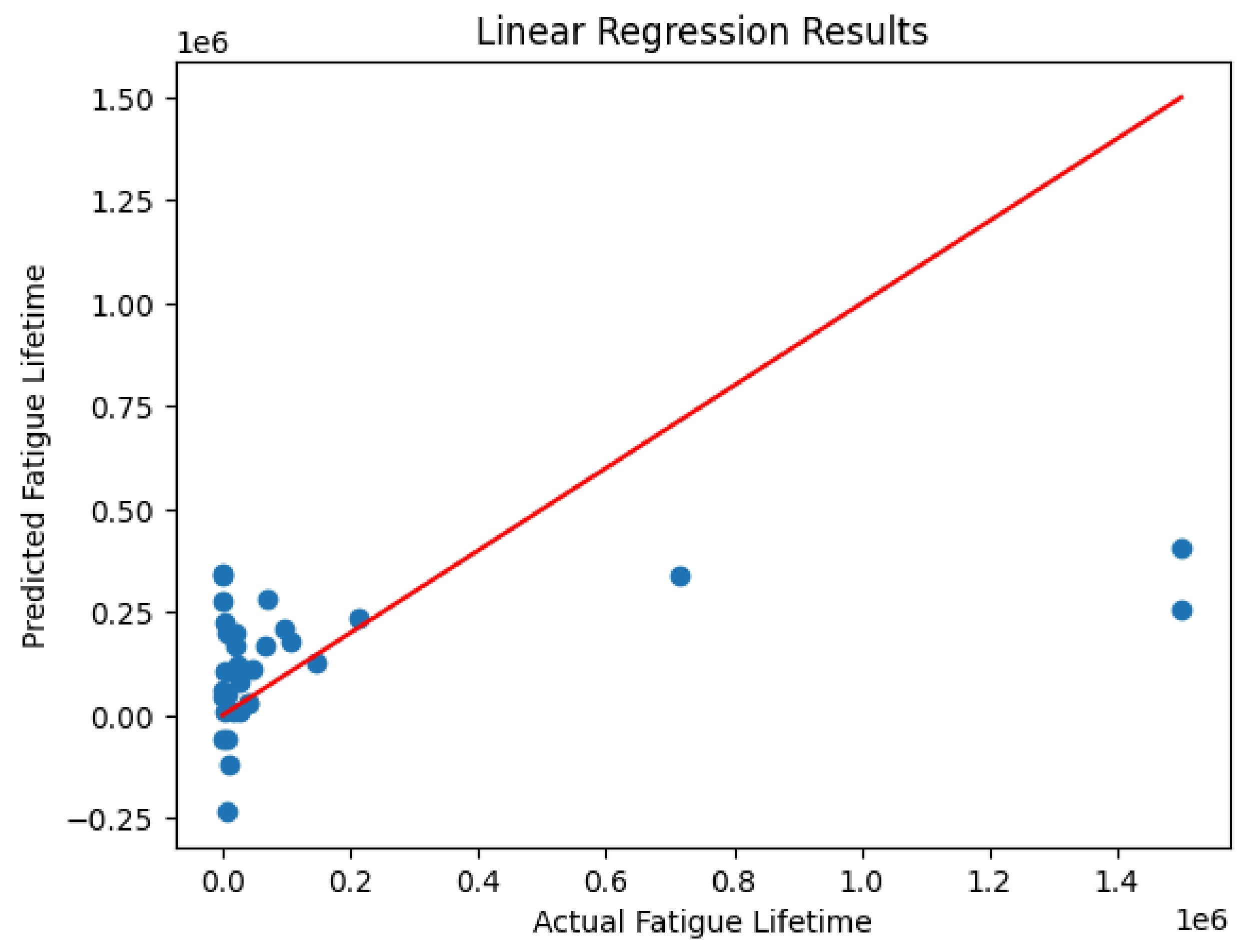


Mean Squared Error: 124554004775.68008
R-squared: 0.022846045412546756

Linear regression method:

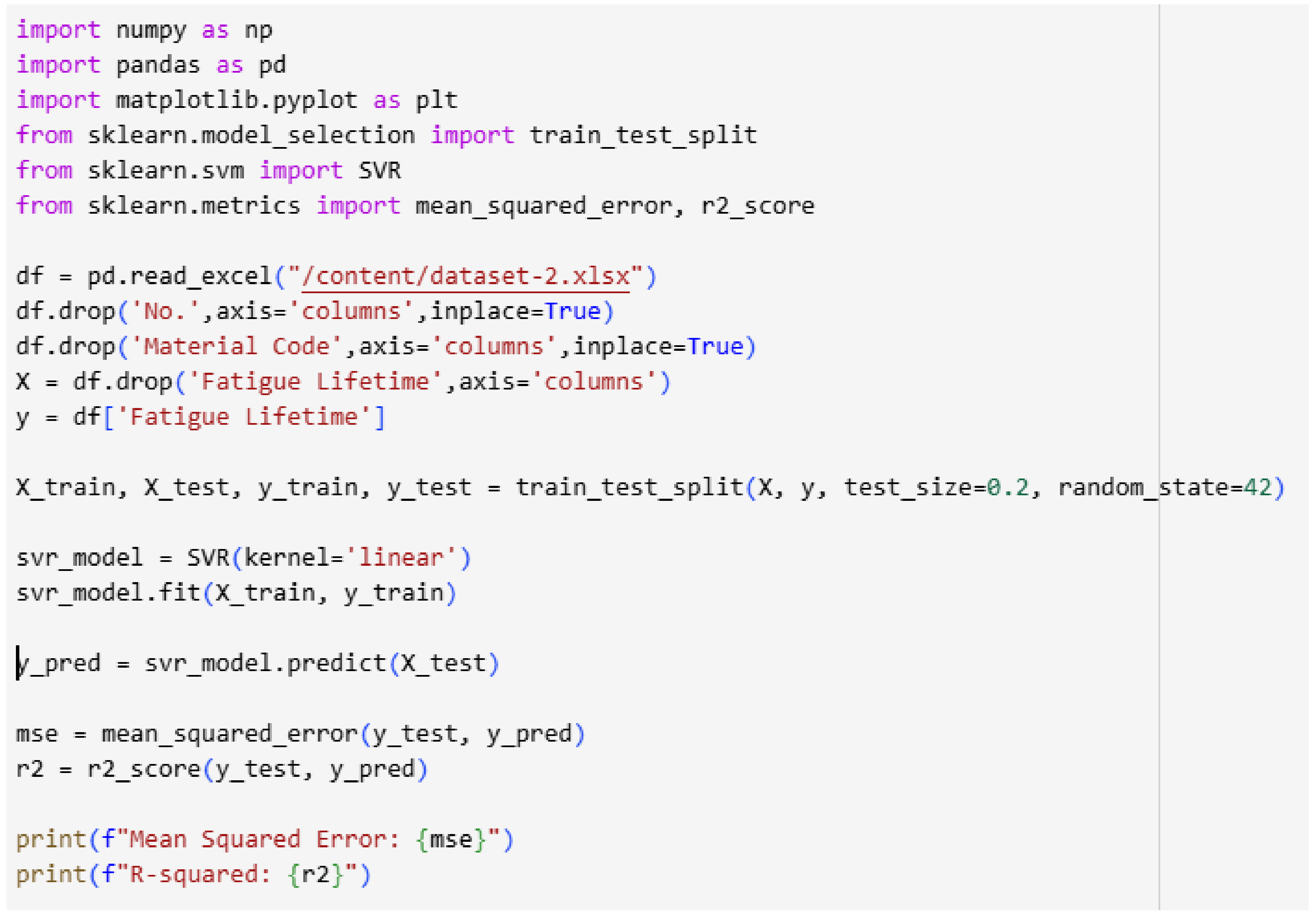


```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_excel("/content/dataset-2.xlsx")
df.drop('No.',axis='columns',inplace=True)
df.drop('Material Code',axis='columns',inplace=True)
X = df.drop('Fatigue Lifetime',axis='columns')
y = df['Fatigue Lifetime']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

knn_regressor = KNeighborsRegressor(n_neighbors=5)
knn_regressor.fit(X_train, y_train)

y_pred = knn_regressor.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```



Mean Squared Error: 102183947760.71452
R-squared: 0.19834413329730094

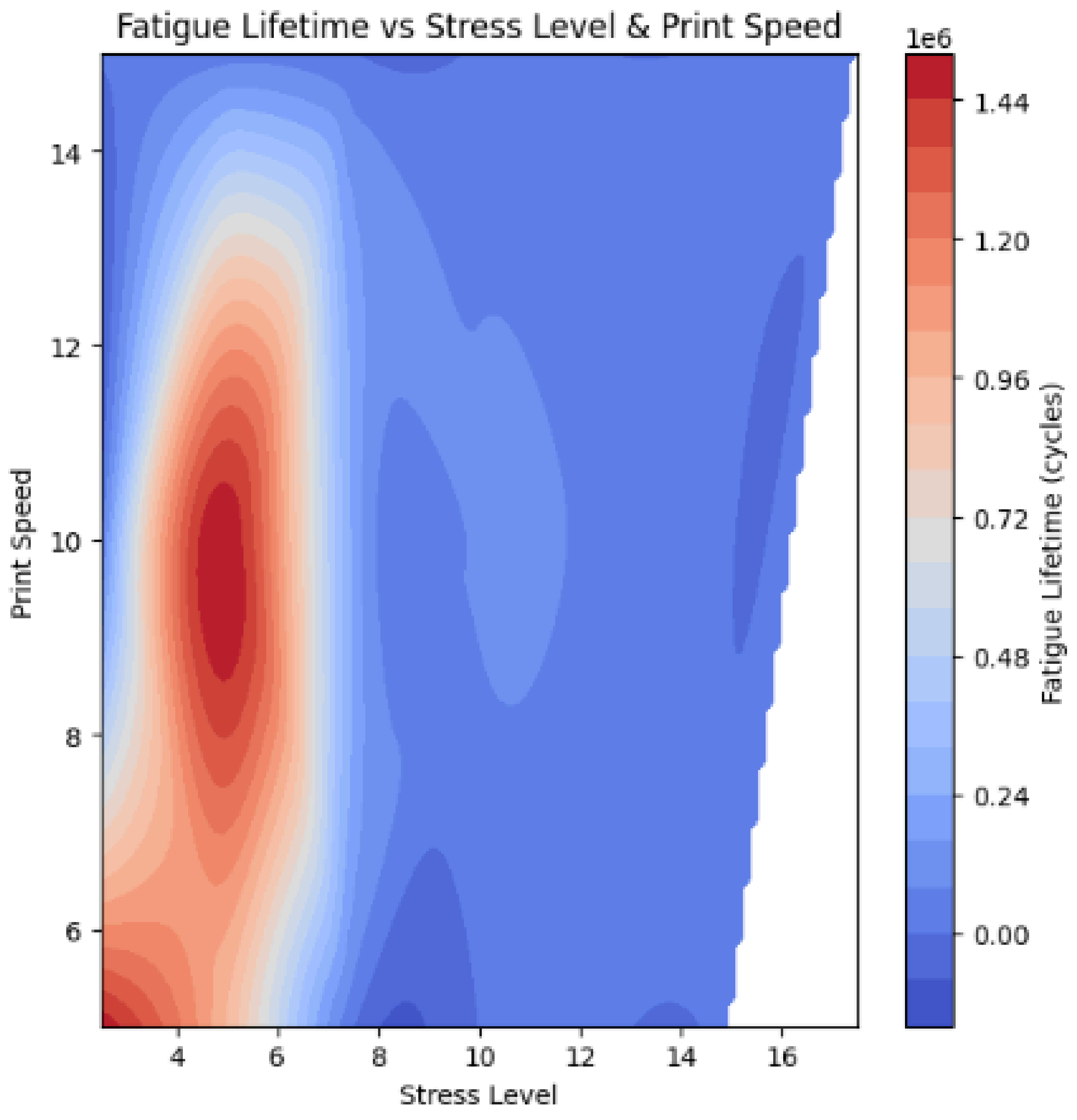
SVM regression method:

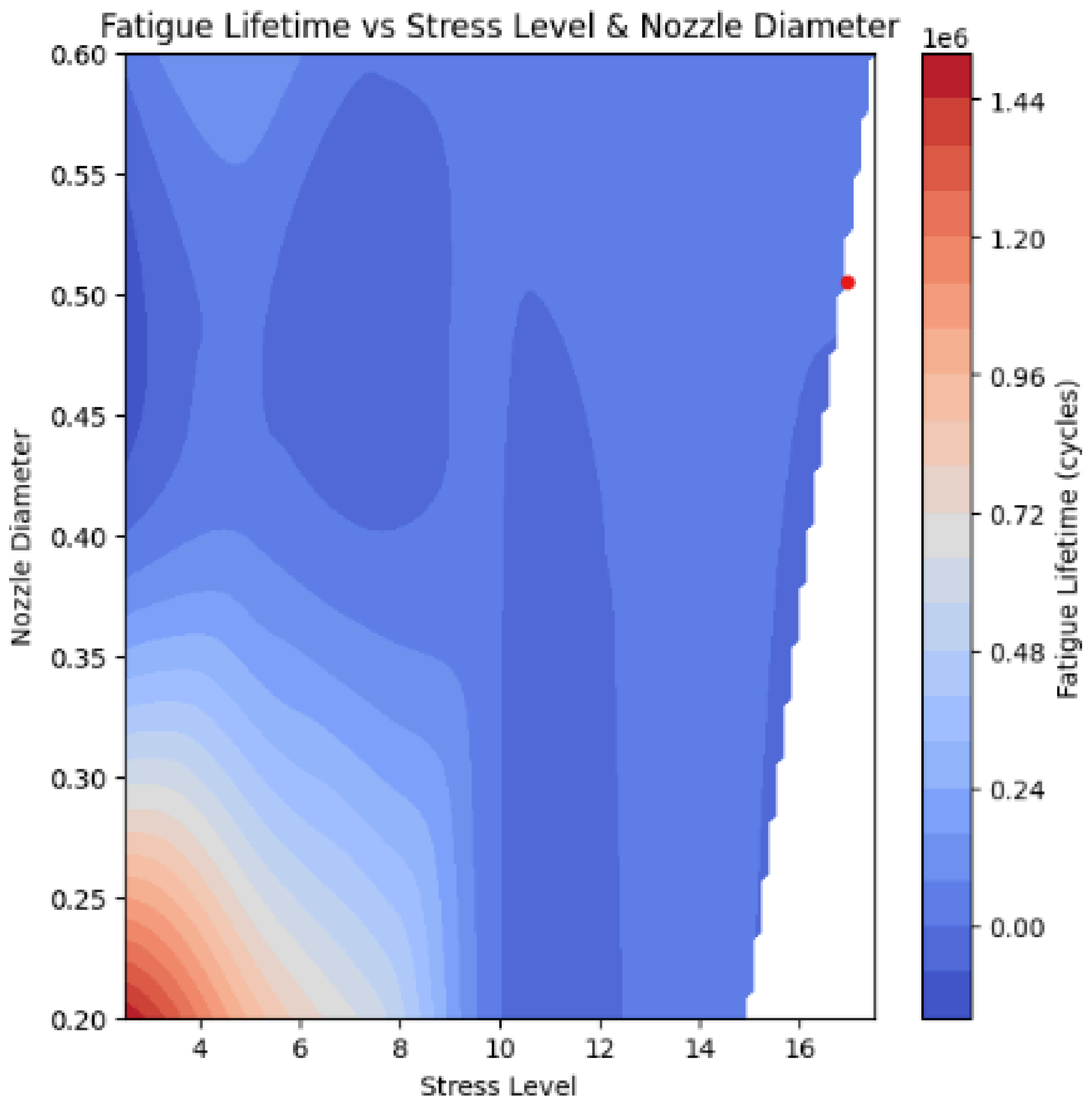


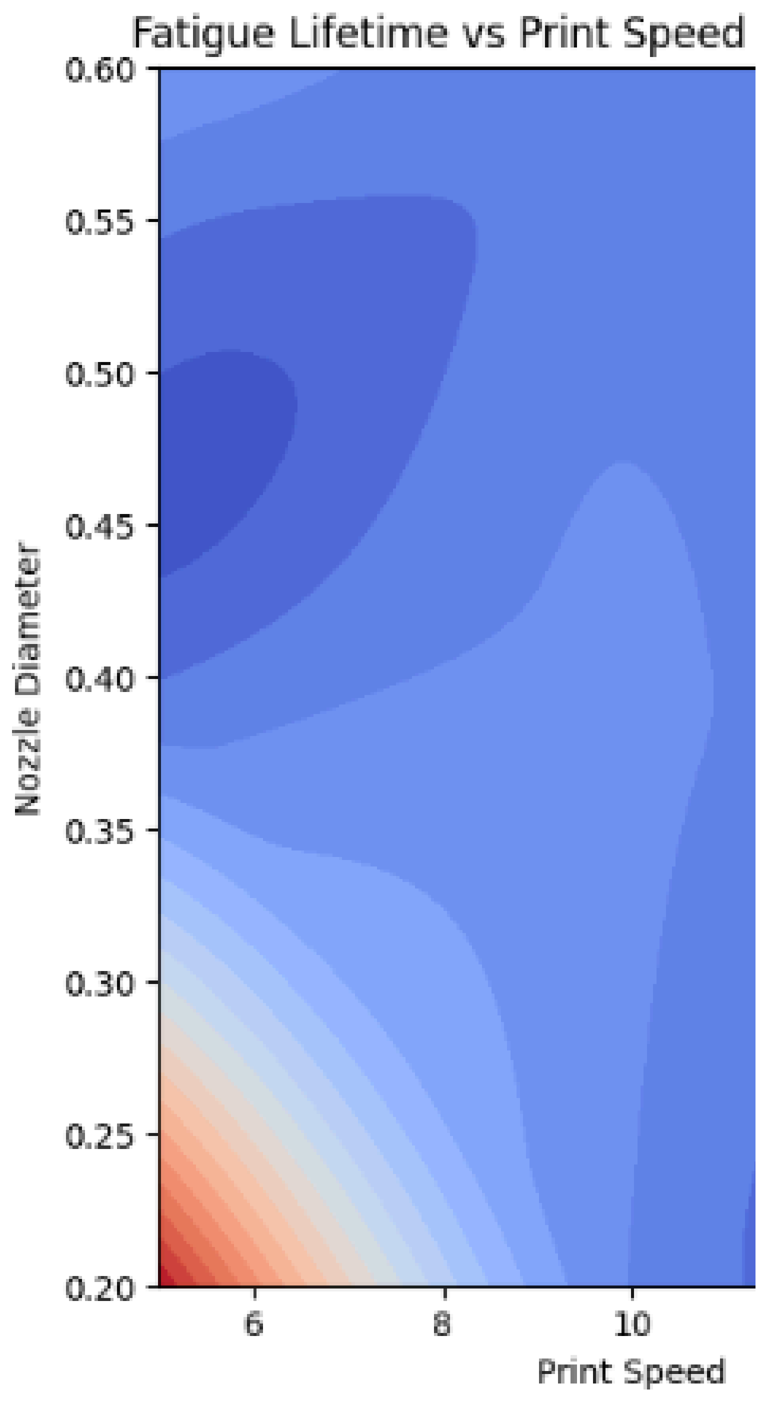
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_excel("/content/dataset-2.xlsx")
df.drop('No.',axis='columns',inplace=True)
df.drop('Material Code',axis='columns',inplace=True)
X = df.drop('Fatigue Lifetime',axis='columns')
y = df['Fatigue Lifetime']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

svr_model = SVR(kernel='linear')
svr_model.fit(X_train, y_train)

y_pred = svr_model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 144302623474.26715
R-squared: -0.13208627405577134
```

Data Characteristics:
Contour Plot:



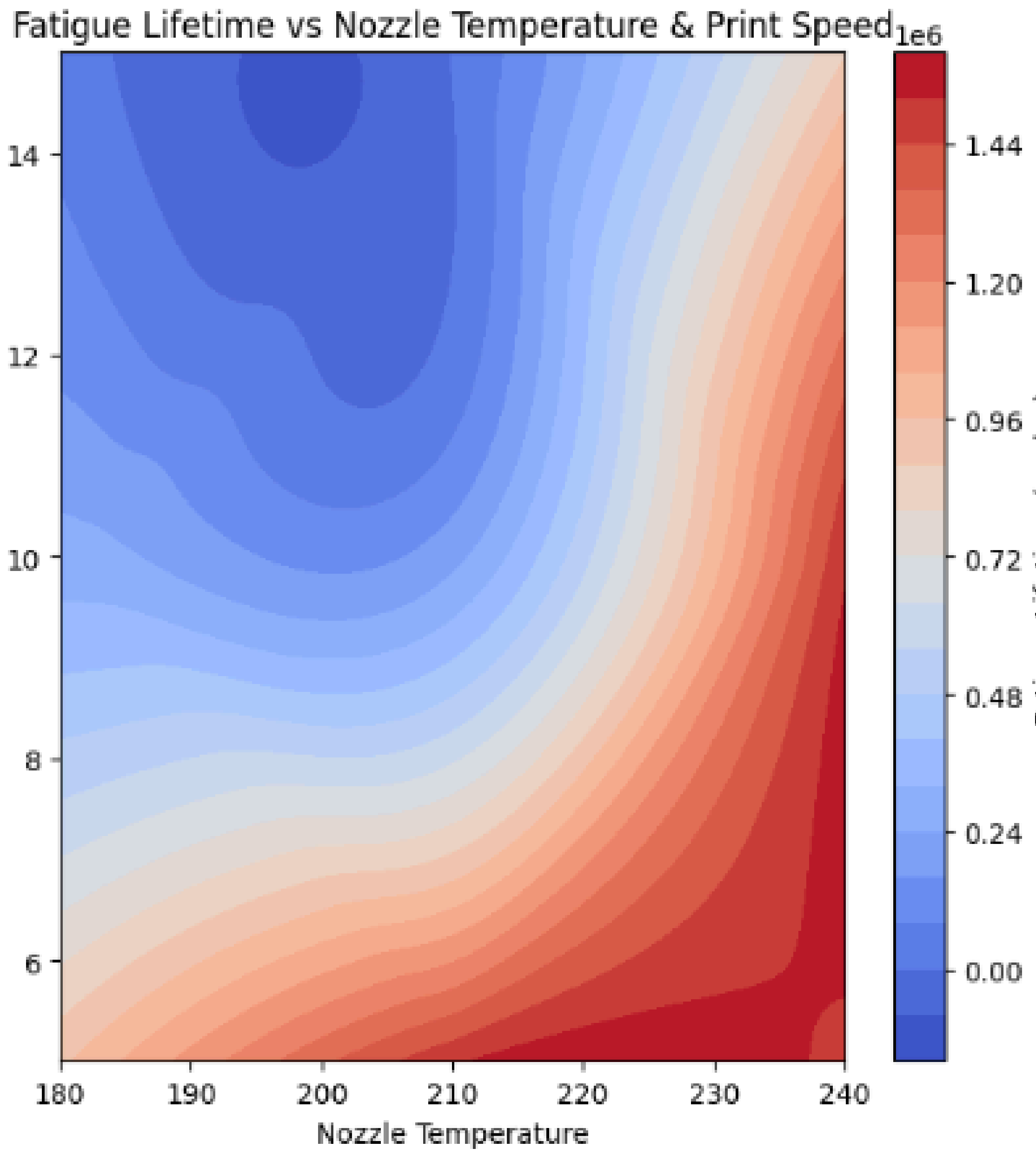


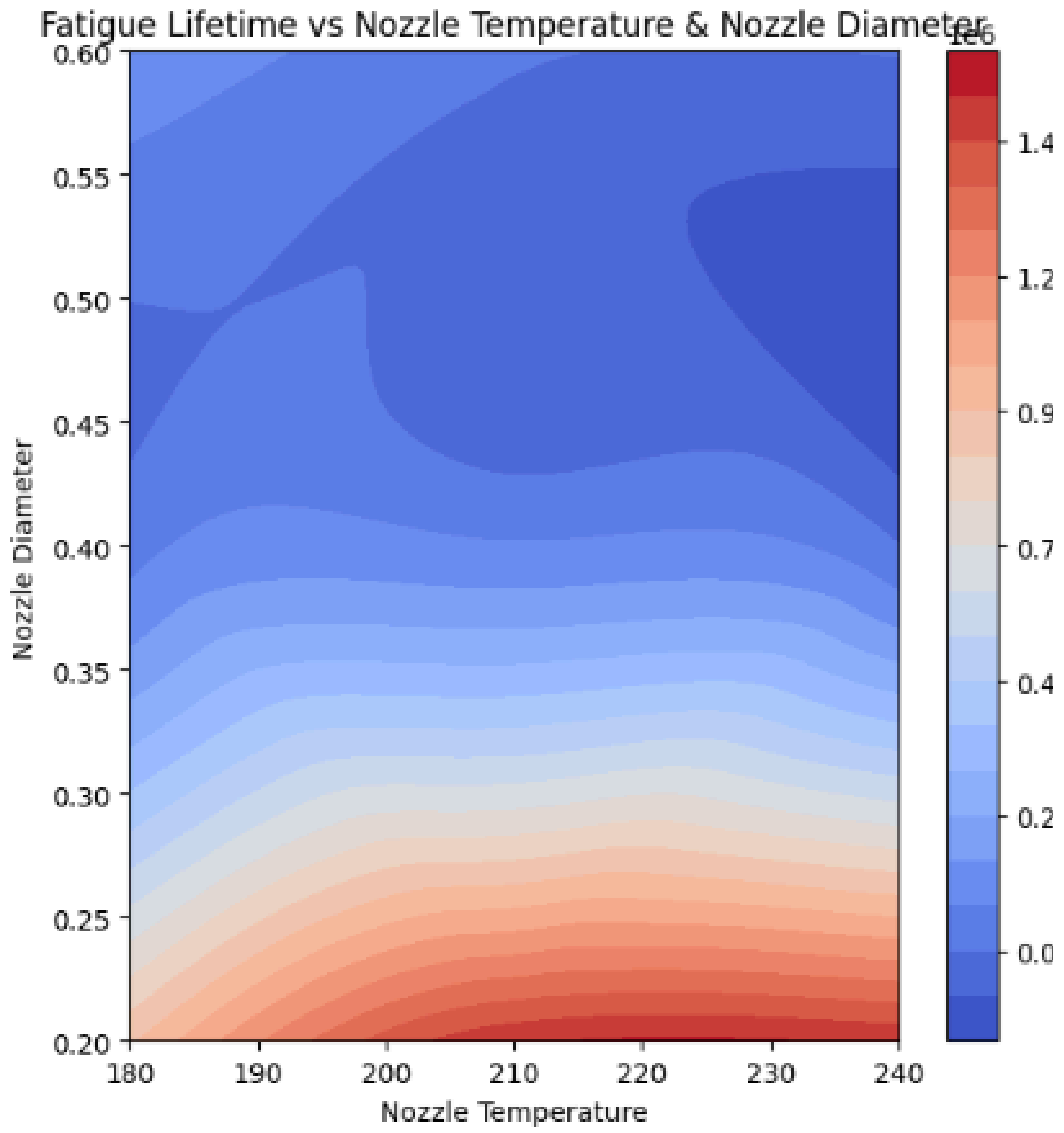


Fatigue Lifetime is better at
Print speed (5 to11) mm/s
Stress level (2.5 to 5) Mpa

Fatigue Lifetime is better at
Nozzle Dia (0.2 to 0.25) mm
Stress level (2.5 to 5) Mpa

Fatigue Lifetime is better at
Print speed (5 to 6) mm/s
Nozzle Dia (0.2 to 0.25) mm





Fatigue lifetime Range is (4e2 to 1.5) e6 Cycles

Print Speed Range is (5 to 15) mm/s

Nozzle Diameter Range is (0.2 to 0.6) mm

Nozzle Temperature Range is (180 to 240) ℃

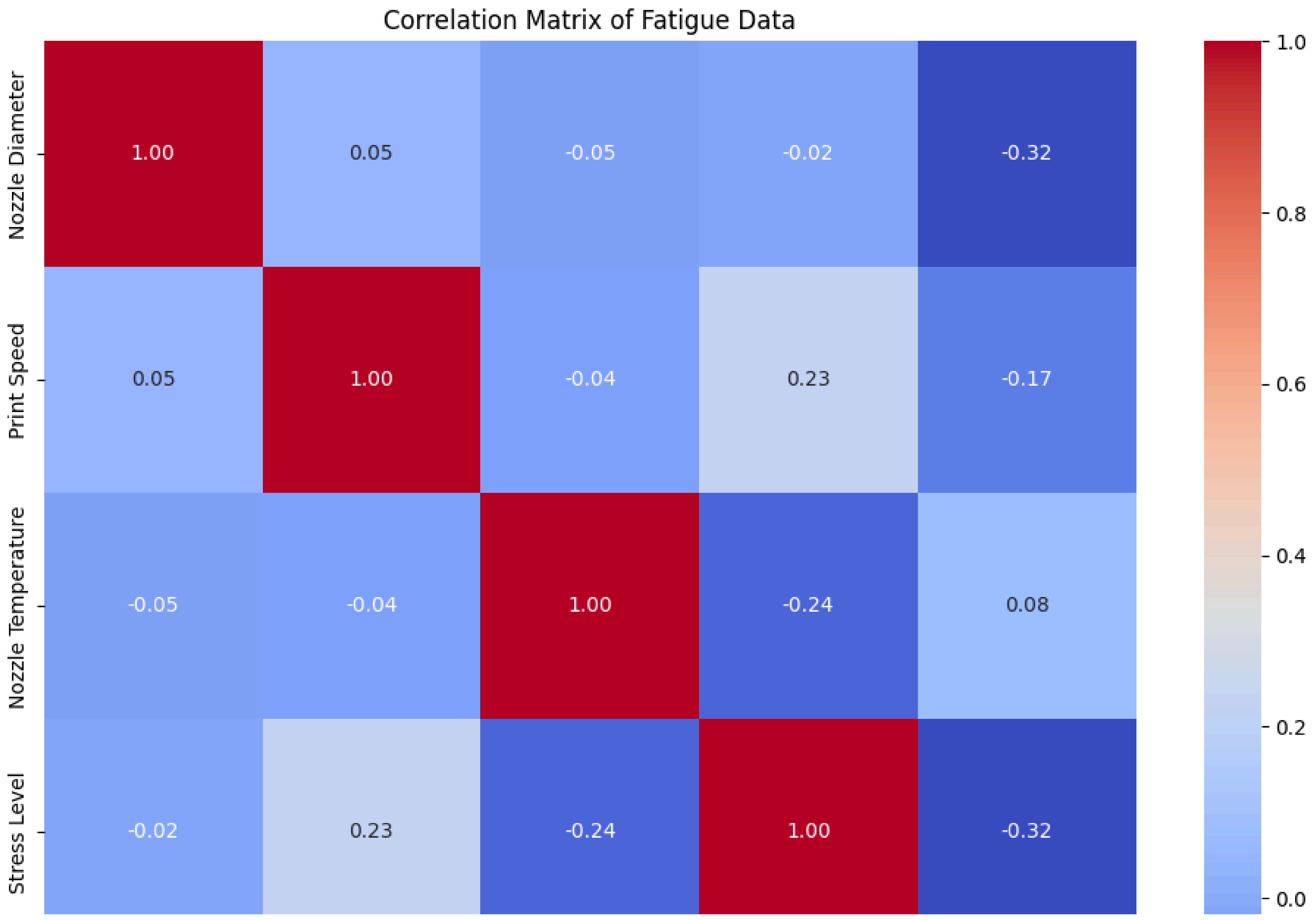
Stress Level Range is (2.5 to 17.5) Mpa

As per the data

Fatigue Lifetime is better at
Print speed (6 to 8) mm/s
Nozzle Temp (220-240)℃

Fatigue Lifetime is better at
Nozzle Dia (0.2 to 0.25) mm
Nozzle Temp (220-240)℃

PEARSON CORRELATION HEATMAP:



About Correlation:

The numbers in the heatmap represent the Pearson correlation coefficients, which range from **-1 to 1**:

- **1.0**: Perfect positive correlation (when one variable increases, the other also increases).
- **0.0**: No correlation.
- **1.0**: Perfect negative correlation (when one variable increases, the other decreases).

The color bar on the right indicates the strength of the correlation:

- **Red (closer to 1.0)**: Strong positive correlation.
- **Blue (closer to -1.0)**: Strong negative correlation.
- **White (closer to 0.0)**: Weak or no correlation.

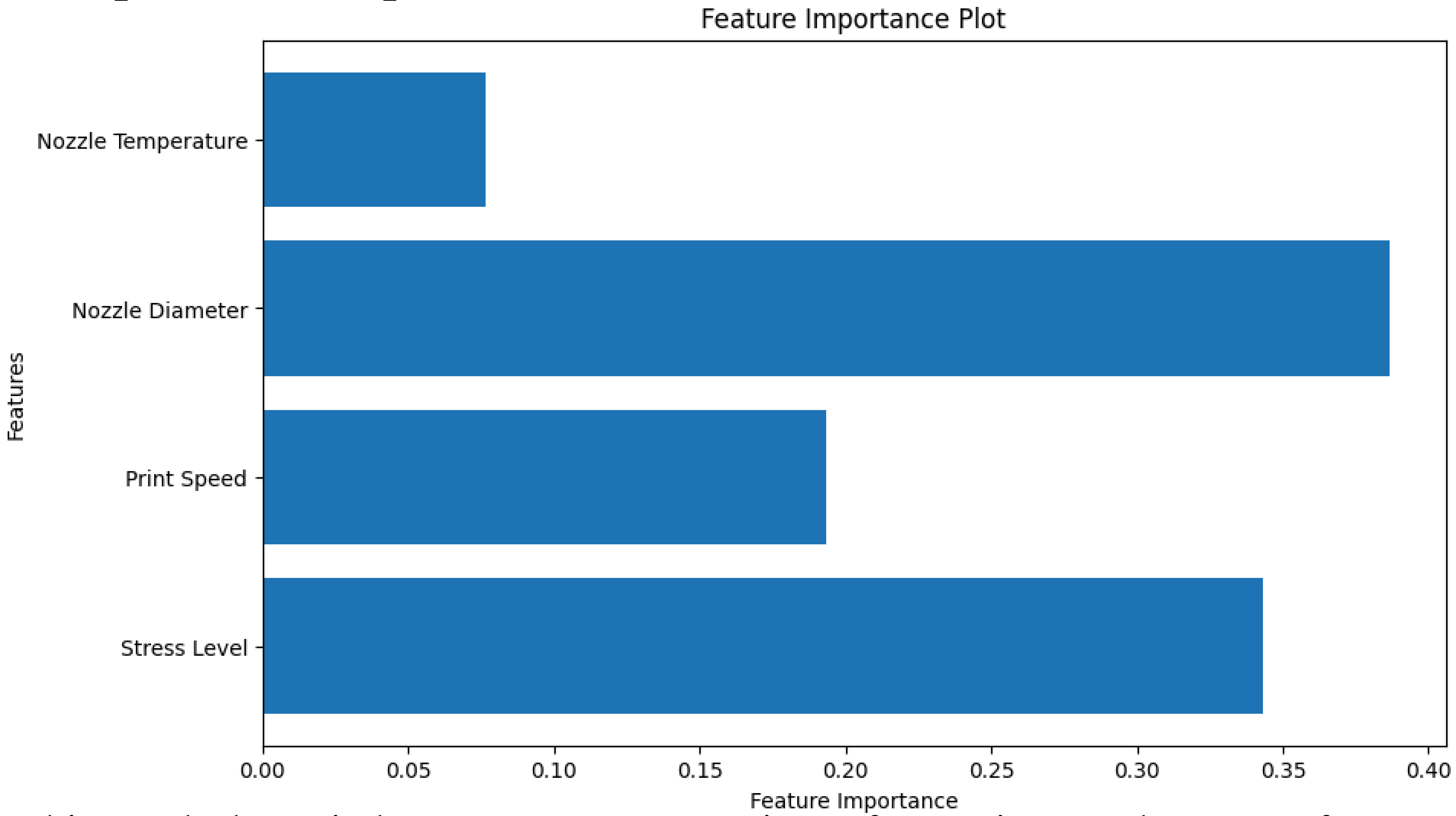
Fatigue Lifetime Trends:

- **Negatively correlated with Nozzle Diameter (-0.32)**: A larger nozzle diameter decreases fatigue lifetime.
- **Negatively correlated with Stress Level (-0.32)**: Higher stress reduces fatigue life.
- **Negatively correlated with Print Speed (-0.17)**: Higher print speeds slightly decrease fatigue life.
- **Weak correlation with Nozzle Temperature (0.08)**: Temperature has little impact on fatigue life.

The most significant correlations are:

- **Stress Level vs. Fatigue Lifetime (-0.32)** → Higher stress reduces fatigue life.
- **Nozzle Diameter vs. Fatigue Lifetime (-0.32)** → Larger nozzle diameters reduce fatigue life.
- Other relationships are weak, meaning they do not significantly impact each other.

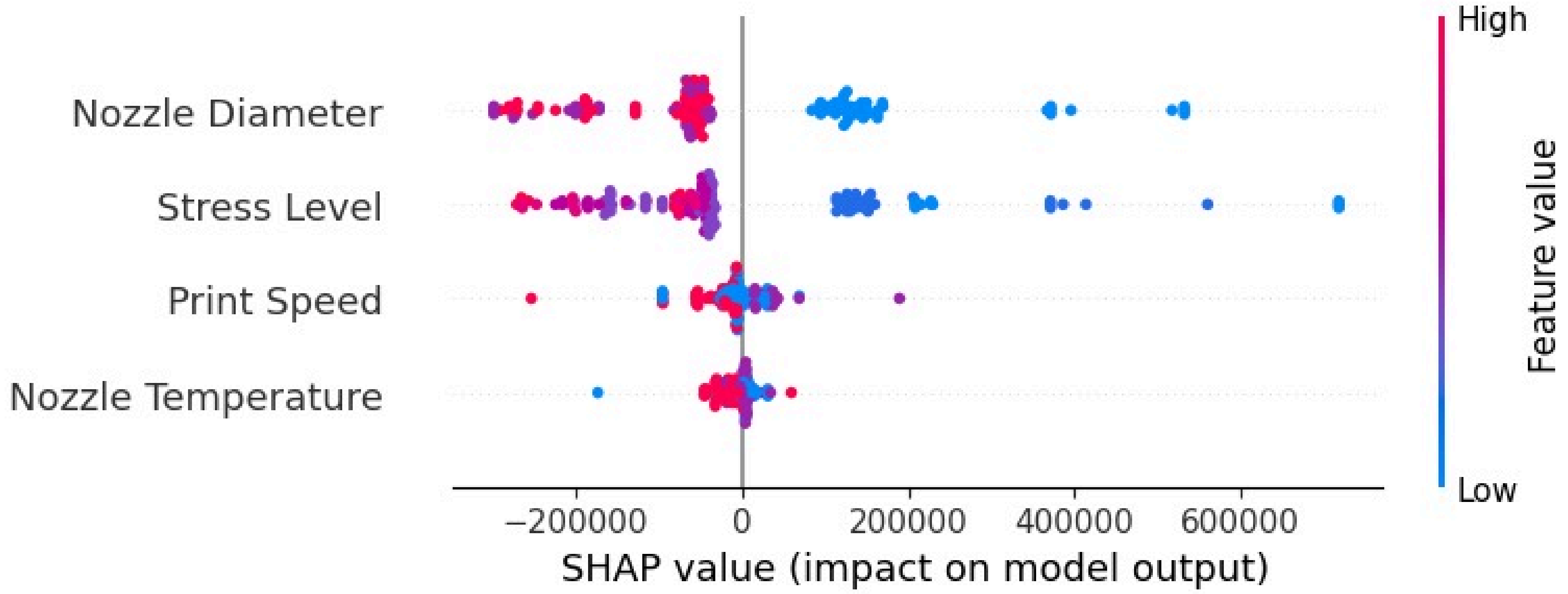
Feature Importance Graph:



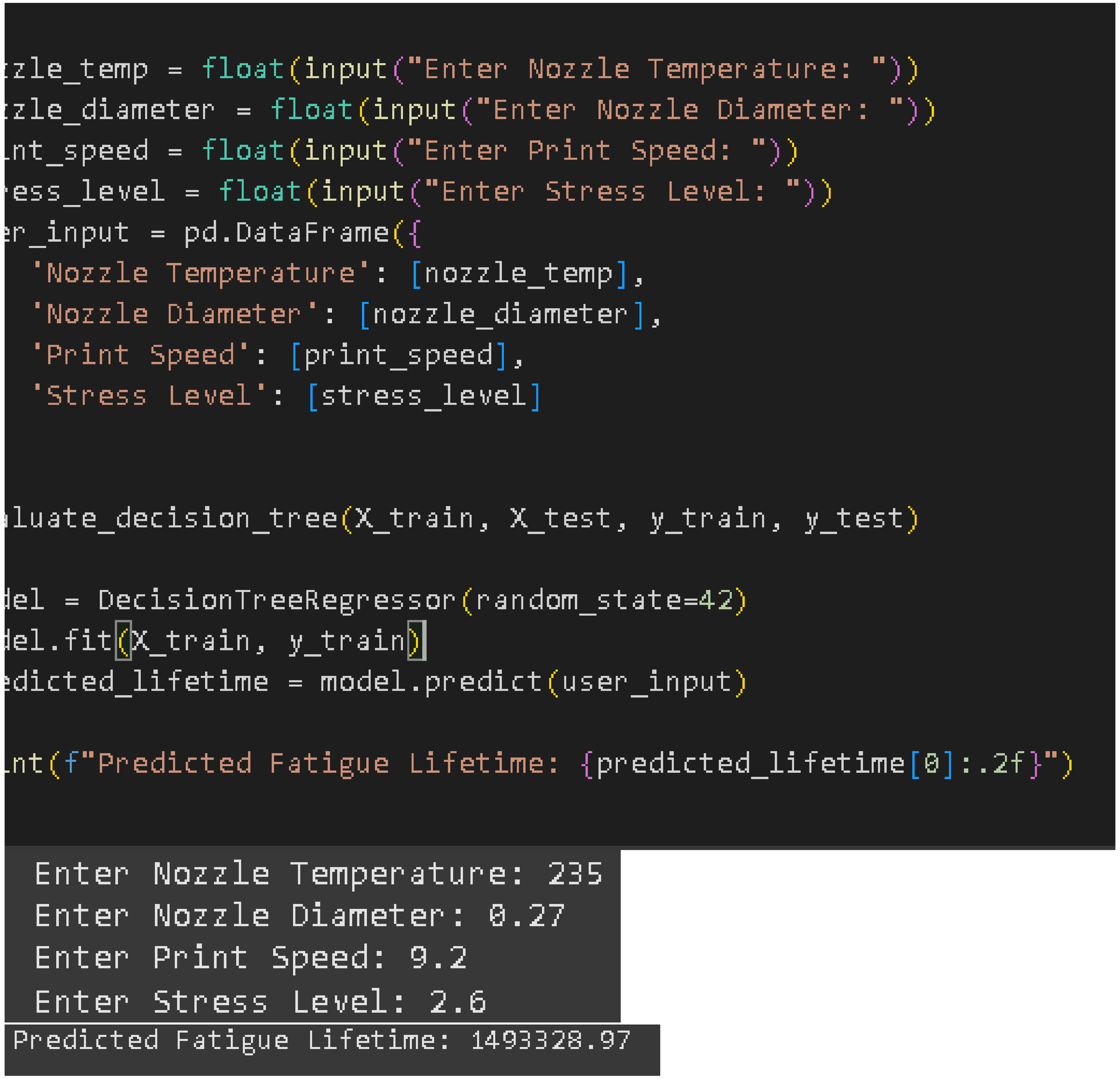
This graph shows in how many percentage input feature impact the output feature:

- Nozzle diameter and Stress level has more percentage influence the output feature (Fatigue Lifetime)

SHAP VALUE EXPLAINATION GRAPH:

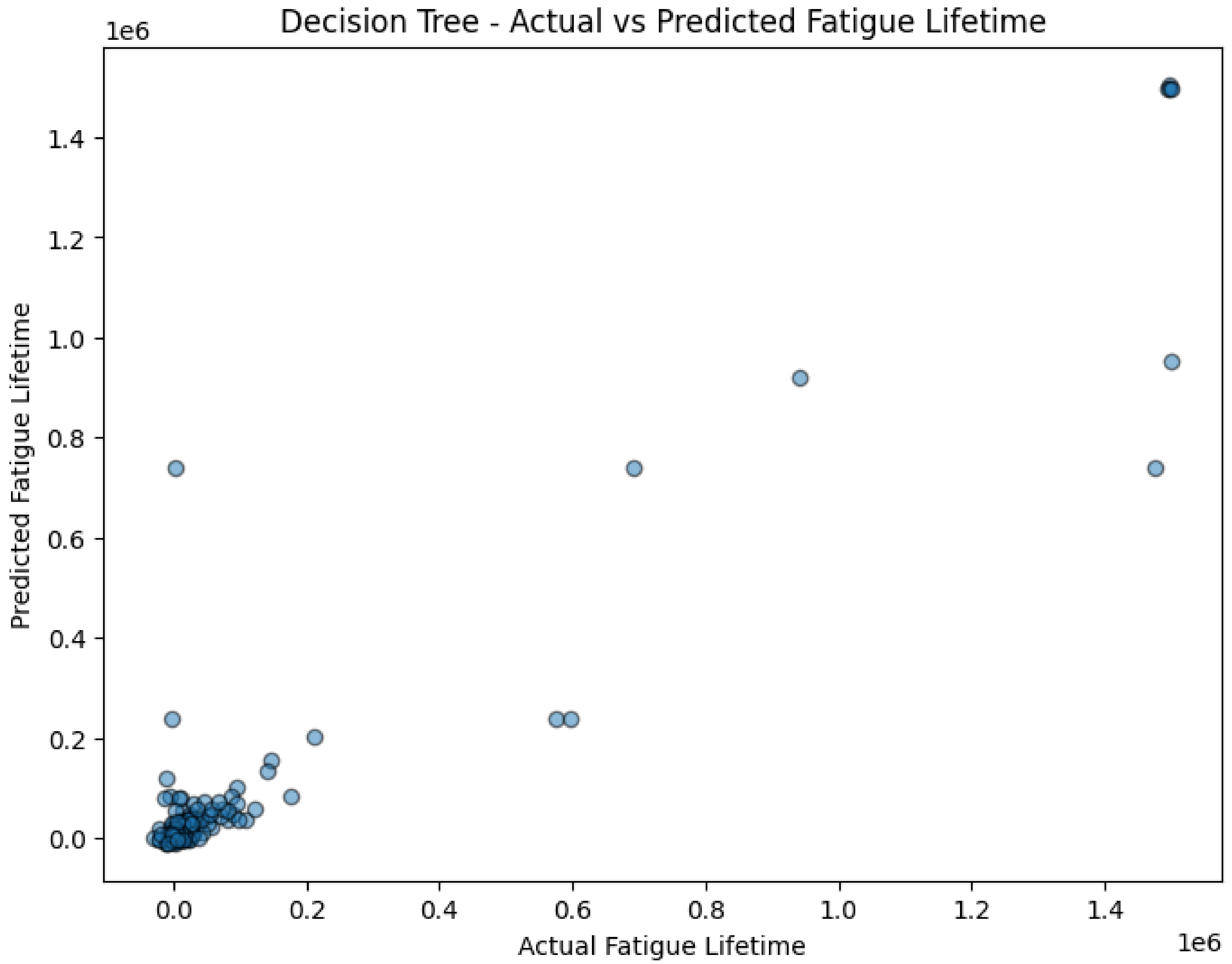


Decision Tree Prediction code:



```
zle_temp = float(input("Enter Nozzle Temperature: "))
zle_diameter = float(input("Enter Nozzle Diameter: "))
nt_speed = float(input("Enter Print Speed: "))
ess_level = float(input("Enter Stress Level: "))
er_input = pd.DataFrame({
  'Nozzle Temperature': [nozzle_temp],
  'Nozzle Diameter': [nozzle_diameter],
  'Print Speed': [print_speed],
  'Stress Level': [stress_level]


luate_decision_tree(X_train, X_test, y_train, y_test)

del = DecisionTreeRegressor(random_state=42)
del.fit(X_train, y_train)
edicted_lifetime = model.predict(user_input)

nt(f"Predicted Fatigue Lifetime: {predicted_lifetime[0]:.2f}")
```

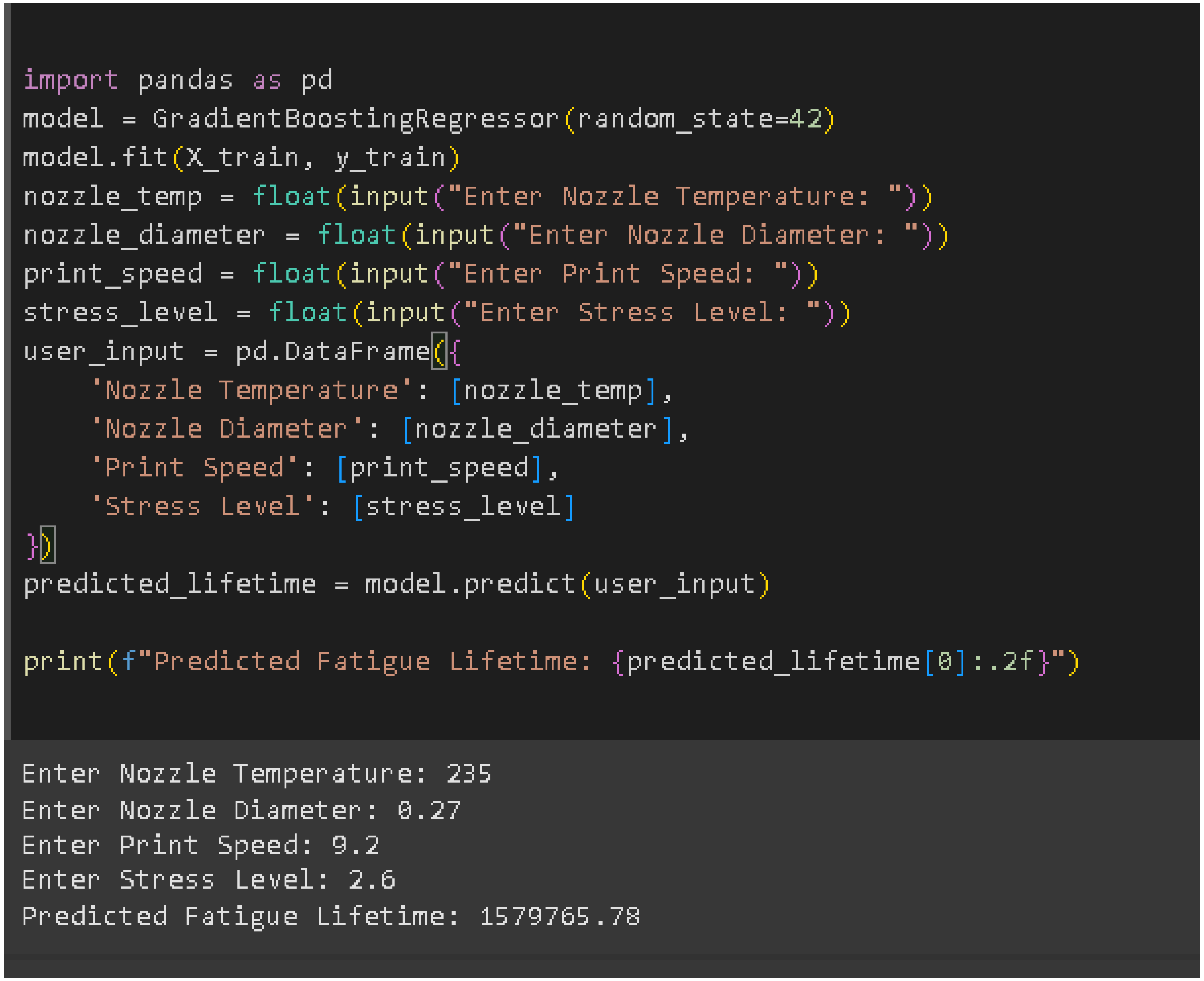
```
Enter Nozzle Temperature: 235
Enter Nozzle Diameter: 0.27
Enter Print Speed: 9.2
Enter Stress Level: 2.6
Predicted Fatigue Lifetime: 1493328.97
```



PREDICTION
Enter Nozzle Temperature: 235
Enter Nozzle Diameter: 0.27
Enter Print Speed: 9.2
Enter Stress Level: 2.6
Predicted Fatigue Lifetime: 1493328.97

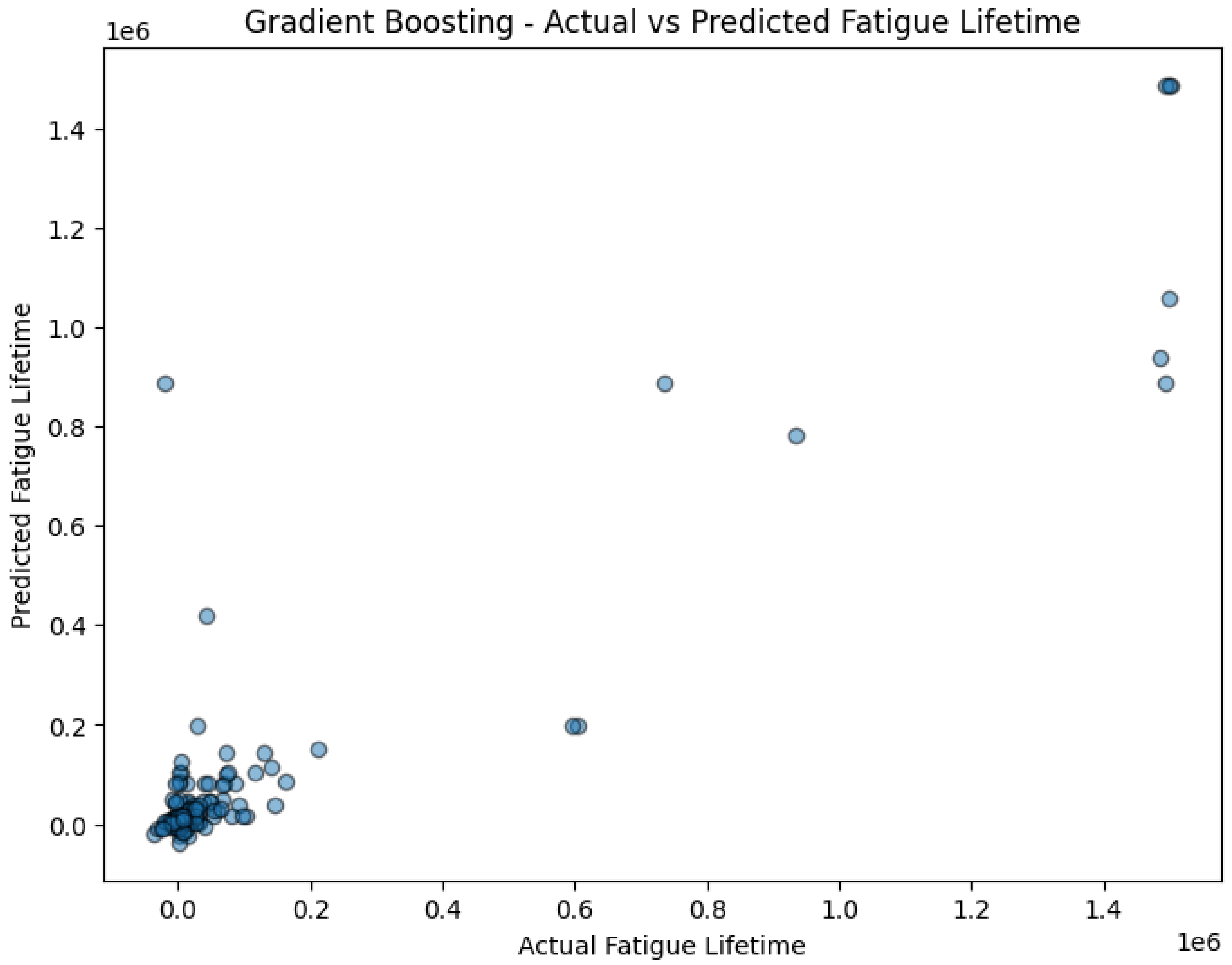
DECISION TREE REGRESSION
R^2: 0.87
RMSE: 12740342474.99
MAE: 38545.05

Gradient Boost Prediction Code:



```
import pandas as pd
model = GradientBoostingRegressor(random_state=42)
model.fit(X_train, y_train)
nozzle_temp = float(input("Enter Nozzle Temperature: "))
nozzle_diameter = float(input("Enter Nozzle Diameter: "))
print_speed = float(input("Enter Print Speed: "))
stress_level = float(input("Enter Stress Level: "))
user_input = pd.DataFrame({
    'Nozzle Temperature': [nozzle_temp],
    'Nozzle Diameter': [nozzle_diameter],
    'Print Speed': [print_speed],
    'Stress Level': [stress_level]
})
predicted_lifetime = model.predict(user_input)

print(f"Predicted Fatigue Lifetime: {predicted_lifetime[0]:.2f}")
```

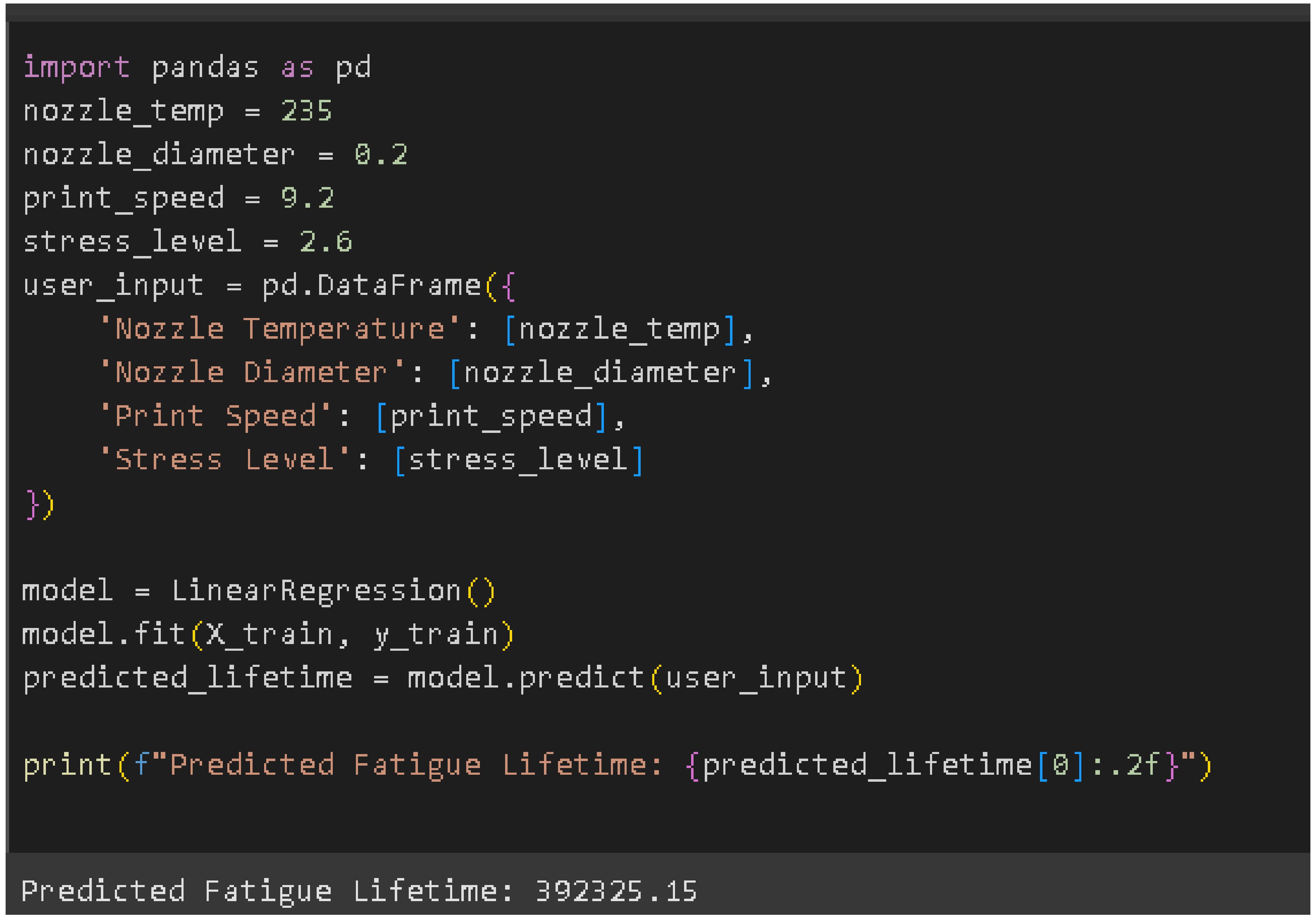
```
Enter Nozzle Temperature: 235
Enter Nozzle Diameter: 0.27
Enter Print Speed: 9.2
Enter Stress Level: 2.6
Predicted Fatigue Lifetime: 1579765.78
```



PREDICTION:
Enter Nozzle Temperature: 235
Enter Nozzle Diameter: 0.27
Enter Print Speed: 9.2
Enter Stress Level: 2.6
Predicted Fatigue Lifetime: 1579765.78

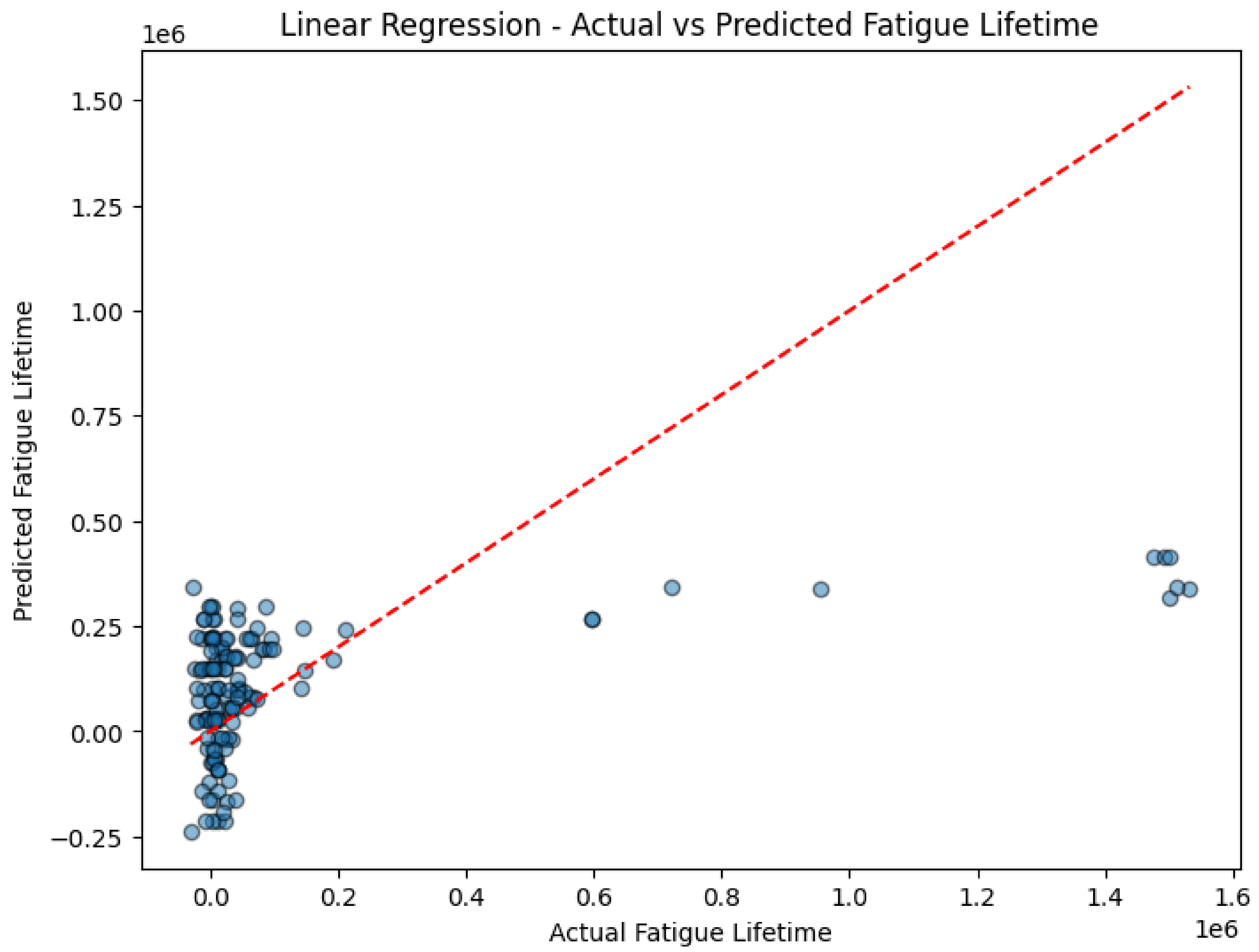
GRADIENT BOOSTING REGRESSION METRICS:
R^2: 0.83
RMSE: 16933231107.53
MAE: 50856.19

Linear Regression Prediction Code:



```python
import pandas as pd
nozzle_temp = 235
nozzle_diameter = 0.2
print_speed = 9.2
stress_level = 2.6
user_input = pd.DataFrame({
    'Nozzle Temperature': [nozzle_temp],
    'Nozzle Diameter': [nozzle_diameter],
    'Print Speed': [print_speed],
    'Stress Level': [stress_level]
})

model = LinearRegression()
model.fit(X_train, y_train)
predicted_lifetime = model.predict(user_input)

print(f"Predicted Fatigue Lifetime: {predicted_lifetime[0]:.2f}")
```

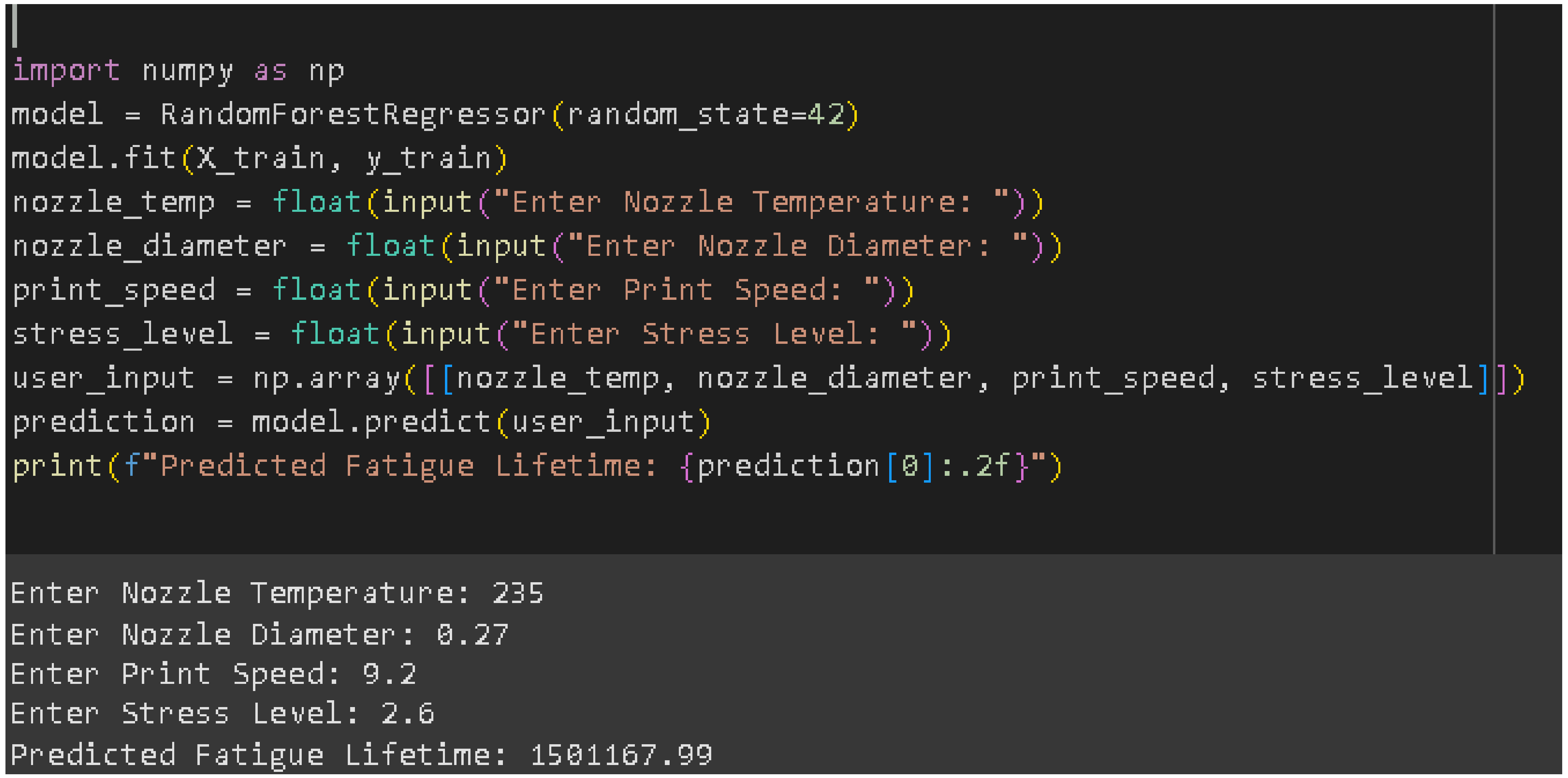
```
Predicted Fatigue Lifetime: 392325.15
```



PREDICTION:
Enter Nozzle Temperature: 235
Enter Nozzle Diameter: 0.27
Enter Print Speed: 9.2
Enter Stress Level: 2.6
Predicted Fatigue Lifetime: 392325.15

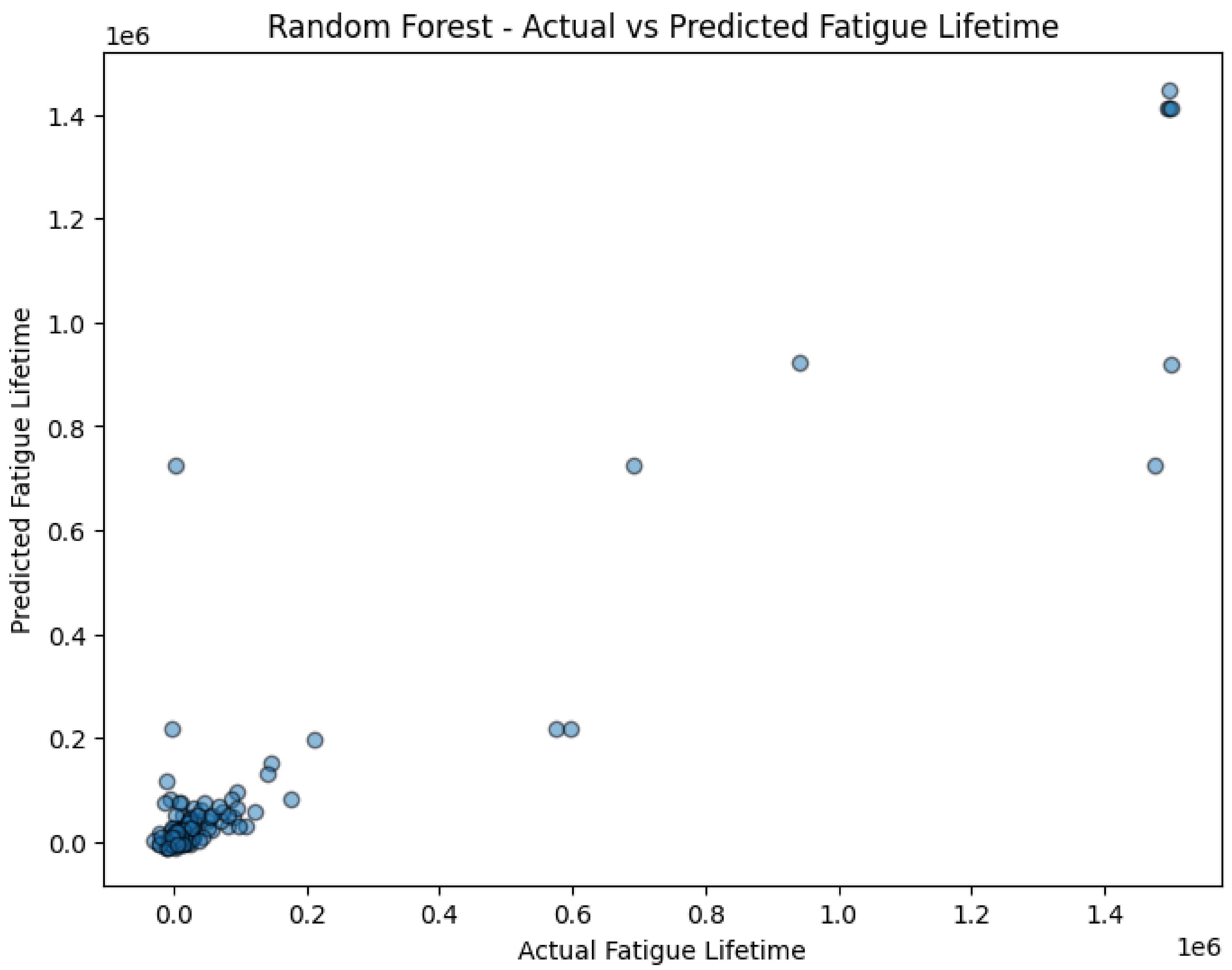
LINEAR REGRESSION METRICS:
R^2: 0.23
RMSE: 79262107356.31
MAE: 170487.64

Random Forest Prediction Code:



```python
import numpy as np
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
nozzle_temp = float(input("Enter Nozzle Temperature: "))
nozzle_diameter = float(input("Enter Nozzle Diameter: "))
print_speed = float(input("Enter Print Speed: "))
stress_level = float(input("Enter Stress Level: "))
user_input = np.array([[nozzle_temp, nozzle_diameter, print_speed, stress_level]])
prediction = model.predict(user_input)
print(f"Predicted Fatigue Lifetime: {prediction[0]:.2f}")
```

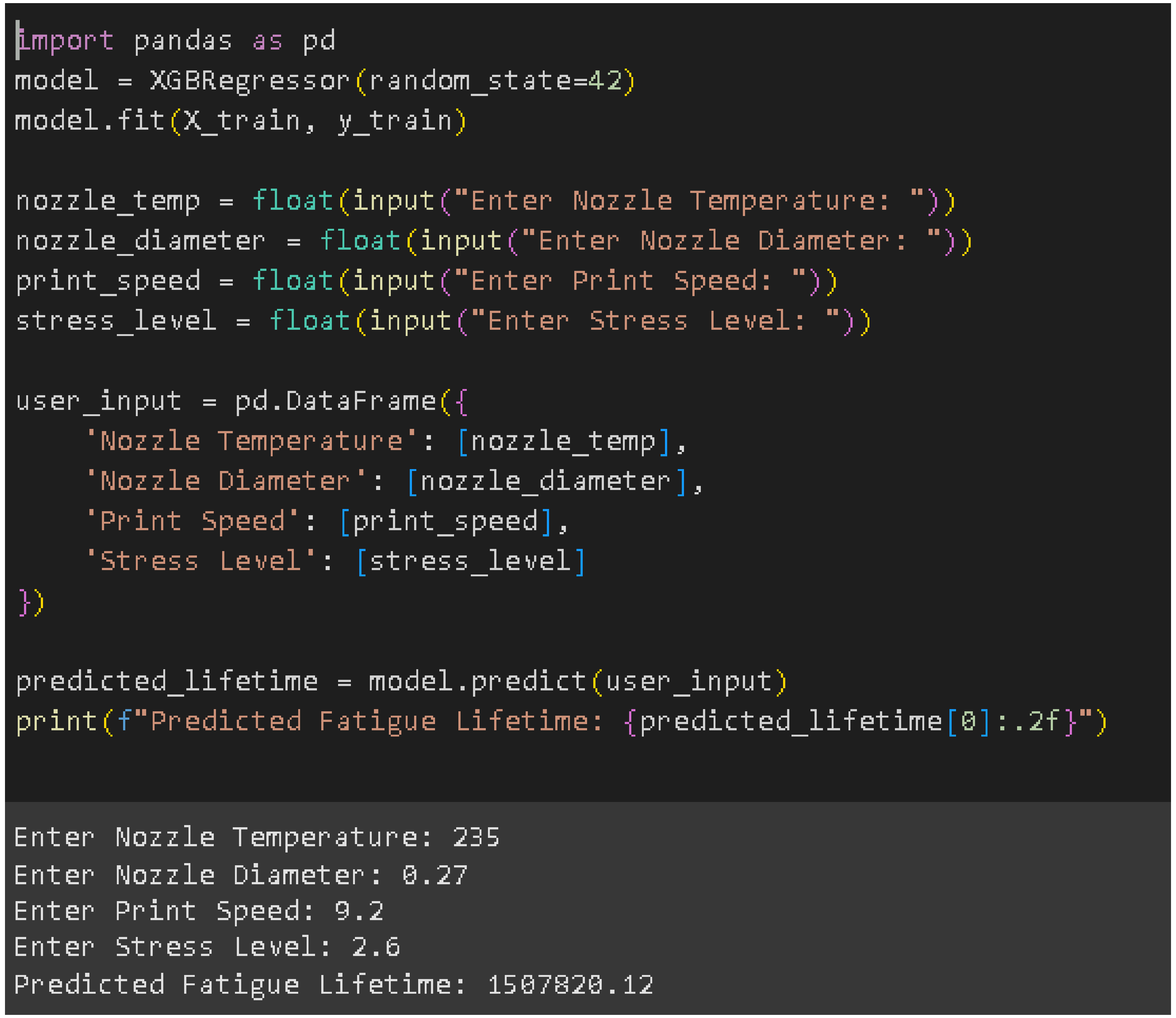
```
Enter Nozzle Temperature: 235
Enter Nozzle Diameter: 0.27
Enter Print Speed: 9.2
Enter Stress Level: 2.6
Predicted Fatigue Lifetime: 1501167.99
```



PREDICTION:
Enter Nozzle Temperature: 235
Enter Nozzle Diameter: 0.27
Enter Print Speed: 9.2
Enter Stress Level: 2.6
Predicted Fatigue Lifetime: 1501167.99

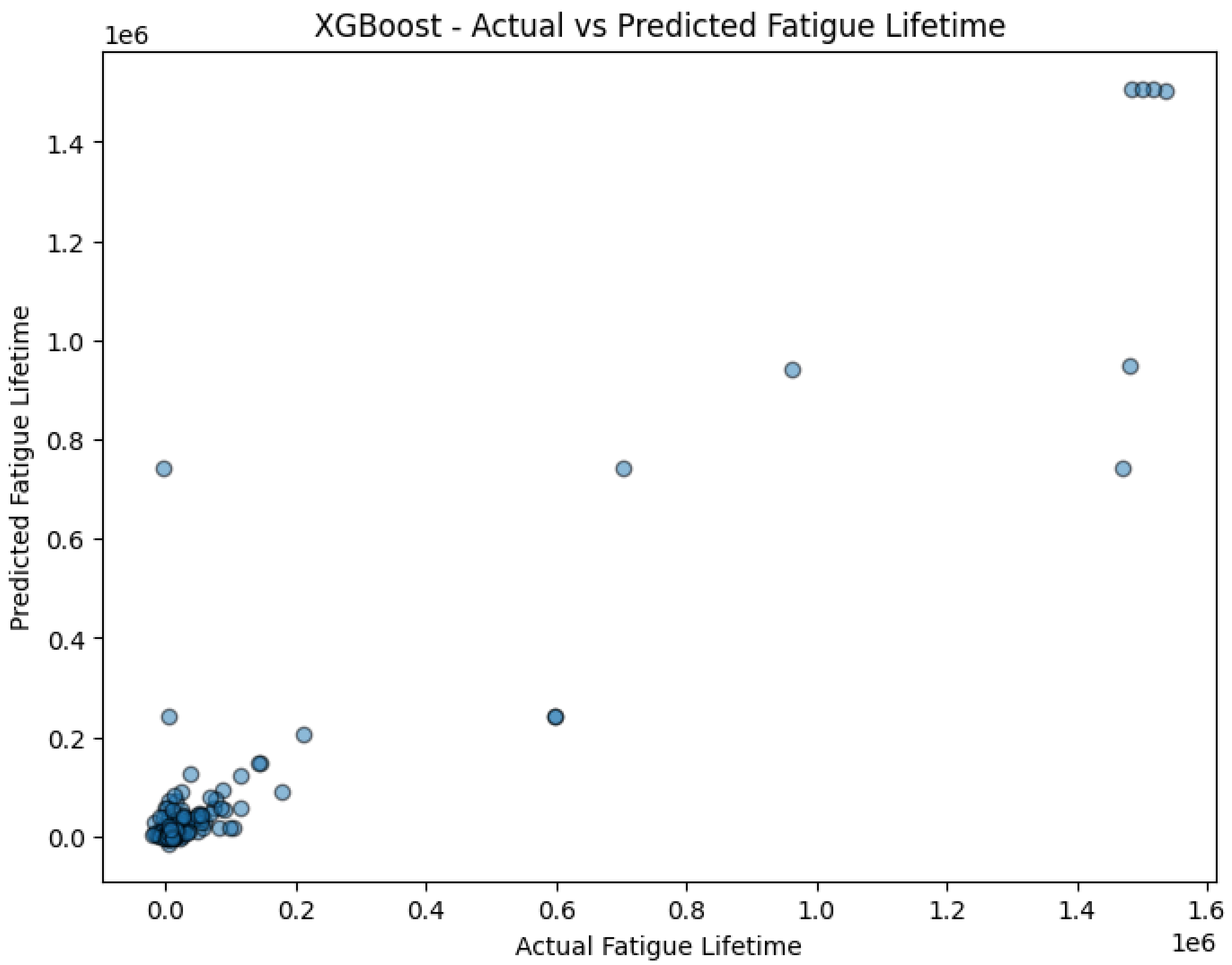
RANDOM FOREST REGRESSION METRICS:
R^2: 0.87
RMSE: 13298799626.90
MAE: 40592.36

XGBOOST PREDICTION CODE:



```python
import pandas as pd
model = XGBRegressor(random_state=42)
model.fit(X_train, y_train)

nozzle_temp = float(input("Enter Nozzle Temperature: "))
nozzle_diameter = float(input("Enter Nozzle Diameter: "))
print_speed = float(input("Enter Print Speed: "))
stress_level = float(input("Enter Stress Level: "))

user_input = pd.DataFrame({
    'Nozzle Temperature': [nozzle_temp],
    'Nozzle Diameter': [nozzle_diameter],
    'Print Speed': [print_speed],
    'Stress Level': [stress_level]
})

predicted_lifetime = model.predict(user_input)
print(f"Predicted Fatigue Lifetime: {predicted_lifetime[0]:.2f}")
```

```
Enter Nozzle Temperature: 235
Enter Nozzle Diameter: 0.27
Enter Print Speed: 9.2
Enter Stress Level: 2.6
Predicted Fatigue Lifetime: 1507820.12
```



PREDICTION:
Enter Nozzle Temperature: 235
Enter Nozzle Diameter: 0.27
Enter Print Speed: 9.2
Enter Stress Level: 2.6
Predicted Fatigue Lifetime: 1507820.12

XGBOOST REGRESSION METRICS:
R^2: 0.88
RMSE: 12668891027.39
MAE: 38498.59

RESULTS AND DISCUSSION:

Comparison of Performance of ML Model:

	BEFORE AUGMENTATION	AFTER AUGMENTATION
ALGORITHM	R^2 VALUE	R^2 VALUE
Decision Tree	-0.69	0.87
Random Forest	-0.47	0.87
Linear Regression	0.19	0.23
Svm	0.13	0.10
KNN	0.02	0.54
Xgboost	-0.69	0.88
Gradient boost	-0.45	0.83

Augmentation: GAUSSIAN NOISE ADDITION

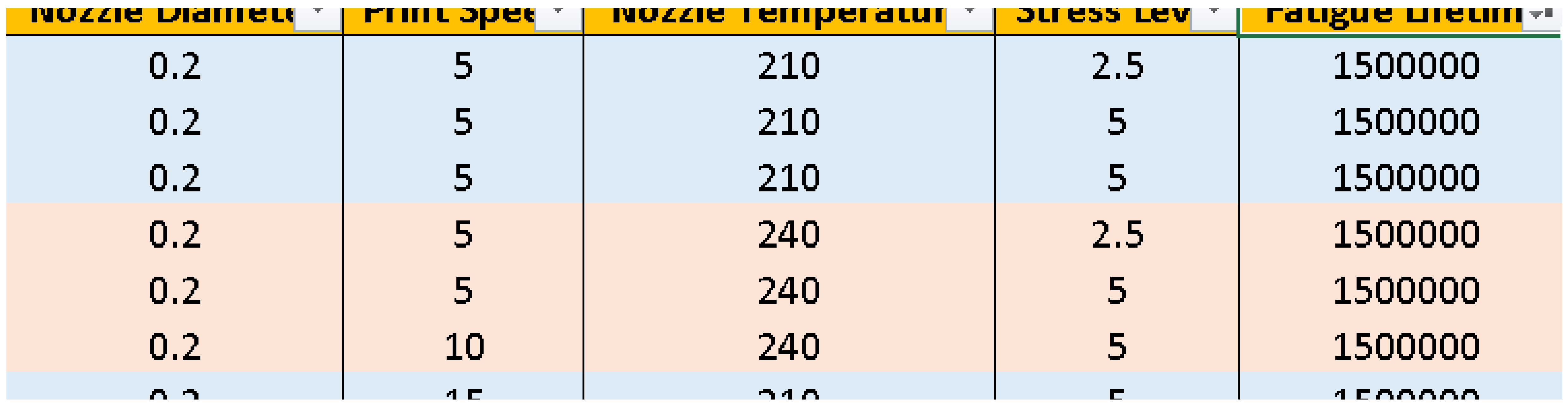
Gaussian Noise Addition refers to adding random noise drawn from a Gaussian (normal) distribution to data. It is commonly used in machine learning and signal processing to improve model robustness, prevent overfitting, or simulate real-world variations. The noise follows a normal distribution defined by a mean (μ) and standard deviation (σ).

Data which is non linear so for linear regression it have low R2 value

Prediction Table:

Prediction Algorithm	Nozzle diameter [mm]	Nozzle temperature [°C]	Stress level [MPa]	Print speed[mm /s]	Predicted fatigue [Cycles]	R^2 Value [0-1]
Linear regression	0.27	235	2.6	9.2	392325.15	0.23
Decision tree	0.27	235	2.6	9.2	1493328.97	0.87
Random Forest	0.27	235	2.6	9.2	1501167.99	0.87
Gradient boost	0.27	235	2.6	9.2	1579765.78	0.83
Xgboost	0.27	235	2.6	9.2	1507820.12	0.88

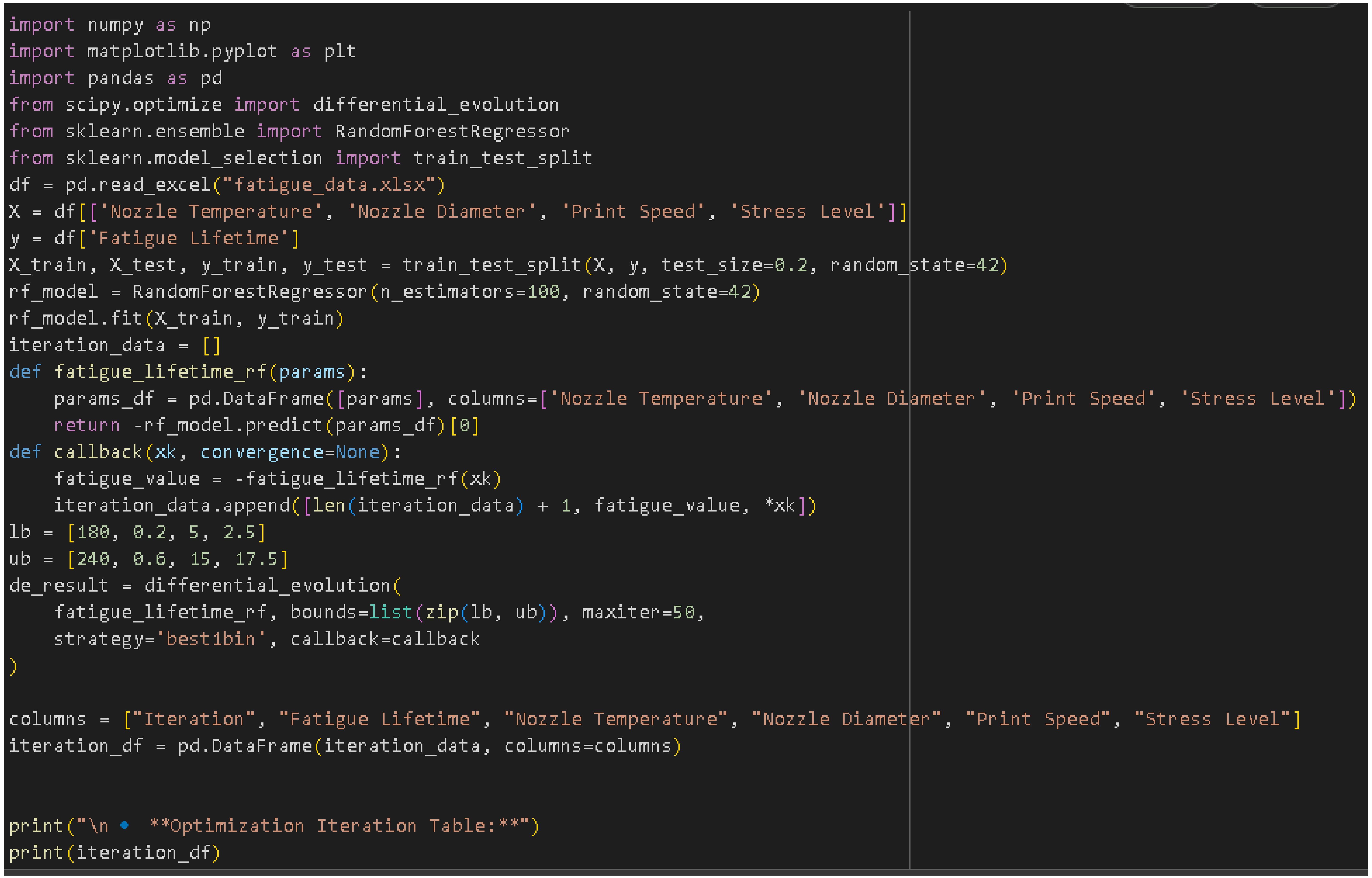
Actual Table:



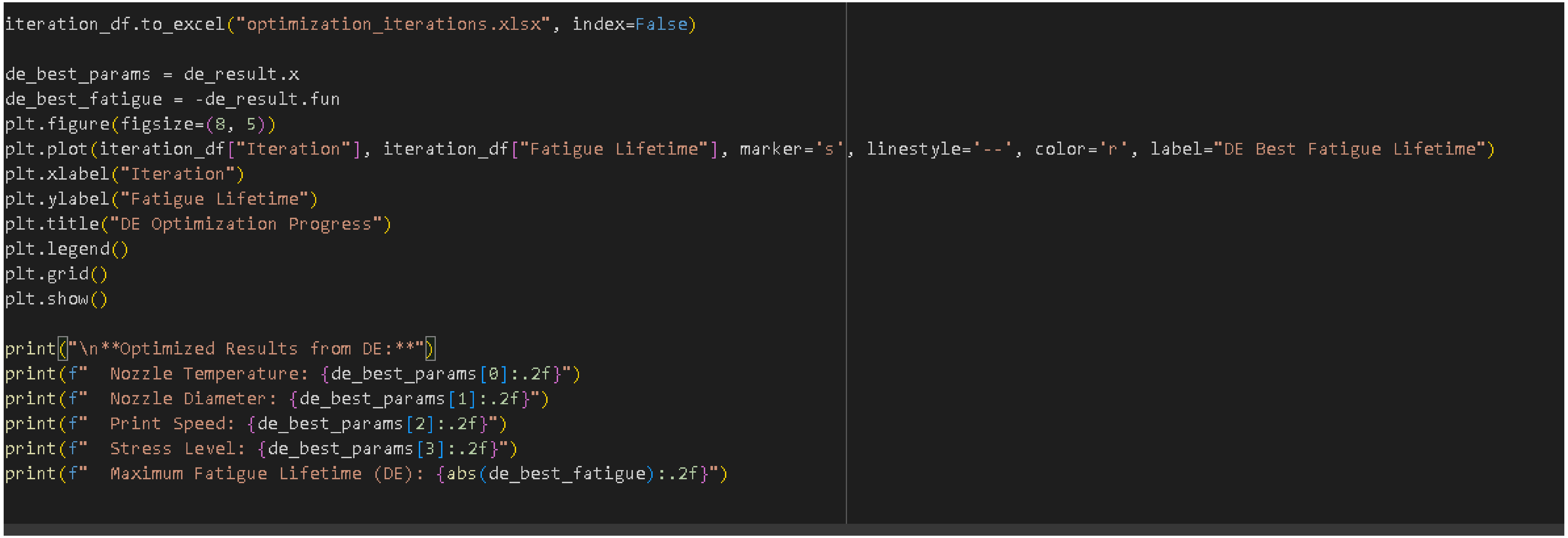
Nozzle Diamete	Print Spee	Nozzle Temperatur	Stress Lev	Fatigue Lifetim
0.2	5	210	2.5	1500000
0.2	5	210	5	1500000
0.2	5	210	5	1500000
0.2	5	240	2.5	1500000
0.2	5	240	5	1500000
0.2	10	240	5	1500000
0.2	15	210	5	1500000

From this R2 value tells data is completely non linear

DIFFERENTIAL EVOLUTION OPTIMIZATION CODE:



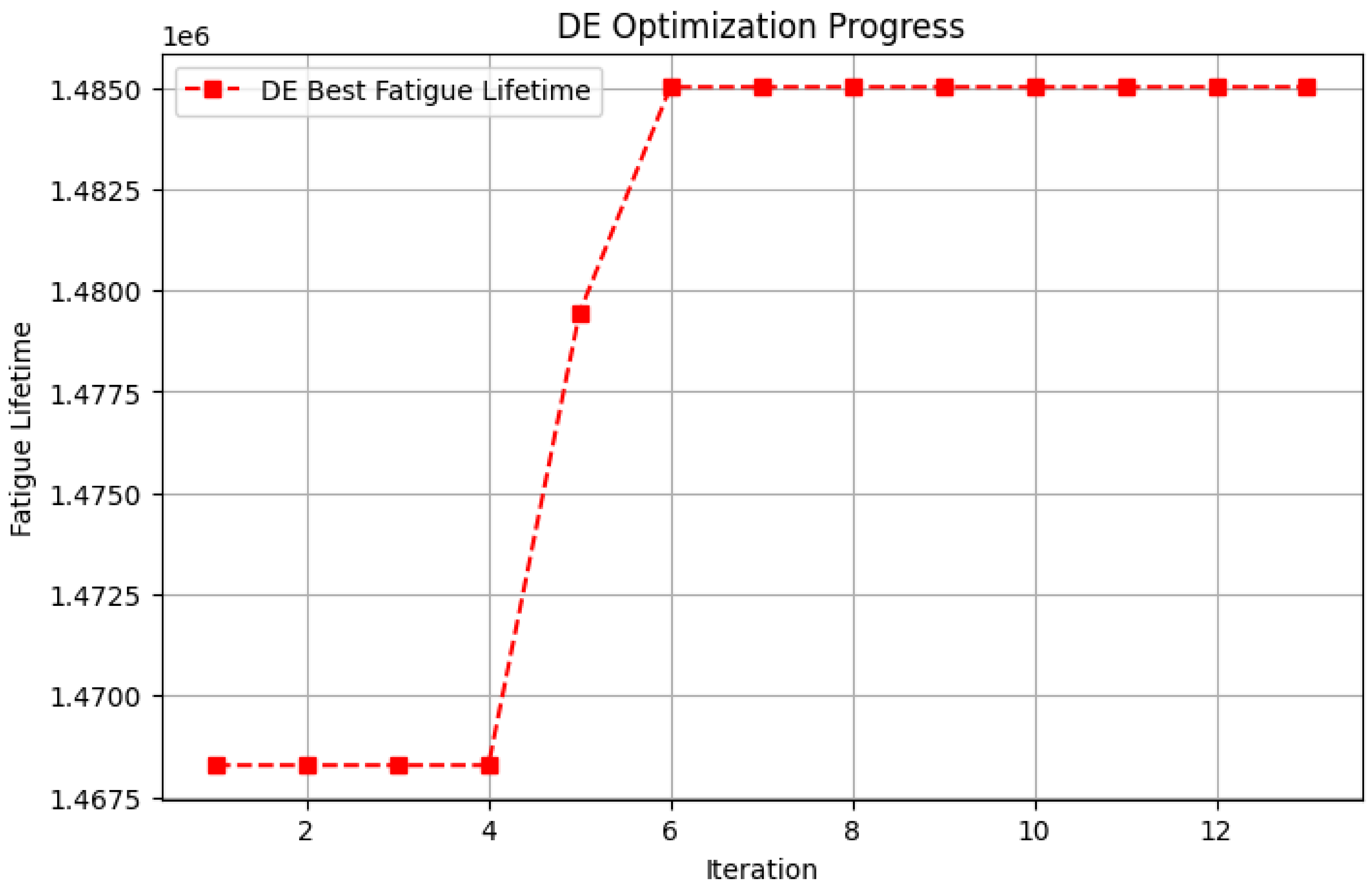
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import differential_evolution
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
df = pd.read_excel("fatigue_data.xlsx")
X = df[['Nozzle Temperature', 'Nozzle Diameter', 'Print Speed', 'Stress Level']]
y = df['Fatigue Lifetime']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
iteration_data = []
def fatigue_lifetime_rf(params):
    params_df = pd.DataFrame([params], columns=['Nozzle Temperature', 'Nozzle Diameter', 'Print Speed', 'Stress Level'])
    return -rf_model.predict(params_df)[0]
def callback(xk, convergence=None):
    fatigue_value = -fatigue_lifetime_rf(xk)
    iteration_data.append([len(iteration_data) + 1, fatigue_value, *xk])
lb = [180, 0.2, 5, 2.5]
ub = [240, 0.6, 15, 17.5]
de_result = differential_evolution(
    fatigue_lifetime_rf, bounds=list(zip(lb, ub)), maxiter=50,
    strategy='best1bin', callback=callback
)

columns = ["Iteration", "Fatigue Lifetime", "Nozzle Temperature", "Nozzle Diameter", "Print Speed", "Stress Level"]
iteration_df = pd.DataFrame(iteration_data, columns=columns)


print("\n 🔹 **Optimization Iteration Table:**")
print(iteration_df)
```



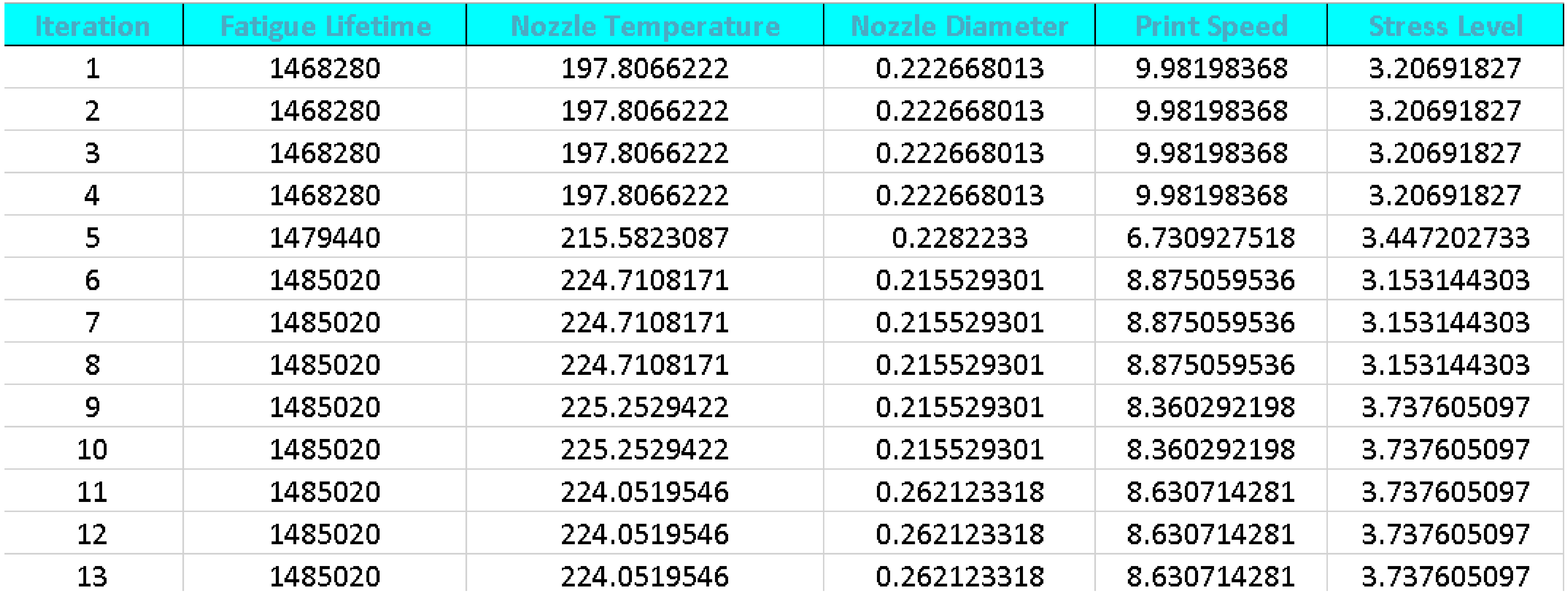
```python
iteration_df.to_excel("optimization_iterations.xlsx", index=False)

de_best_params = de_result.x
de_best_fatigue = -de_result.fun
plt.figure(figsize=(8, 5))
plt.plot(iteration_df["Iteration"], iteration_df["Fatigue Lifetime"], marker='s', linestyle='--', color='r', label="DE Best Fatigue Lifetime")
plt.xlabel("Iteration")
plt.ylabel("Fatigue Lifetime")
plt.title("DE Optimization Progress")
plt.legend()
plt.grid()
plt.show()

print("\n**Optimized Results from DE:**")
print(f"  Nozzle Temperature: {de_best_params[0]:.2f}")
print(f"  Nozzle Diameter: {de_best_params[1]:.2f}")
print(f"  Print Speed: {de_best_params[2]:.2f}")
print(f"  Stress Level: {de_best_params[3]:.2f}")
print(f"  Maximum Fatigue Lifetime (DE): {abs(de_best_fatigue):.2f}")
```

```
Optimized Results from DE:
  Nozzle Temperature: 224.05
  Nozzle Diameter: 0.26
  Print Speed: 8.63
  Stress Level: 3.74
  Maximum Fatigue Lifetime (DE): 1485020.00
```

Differential Evolution Plot:

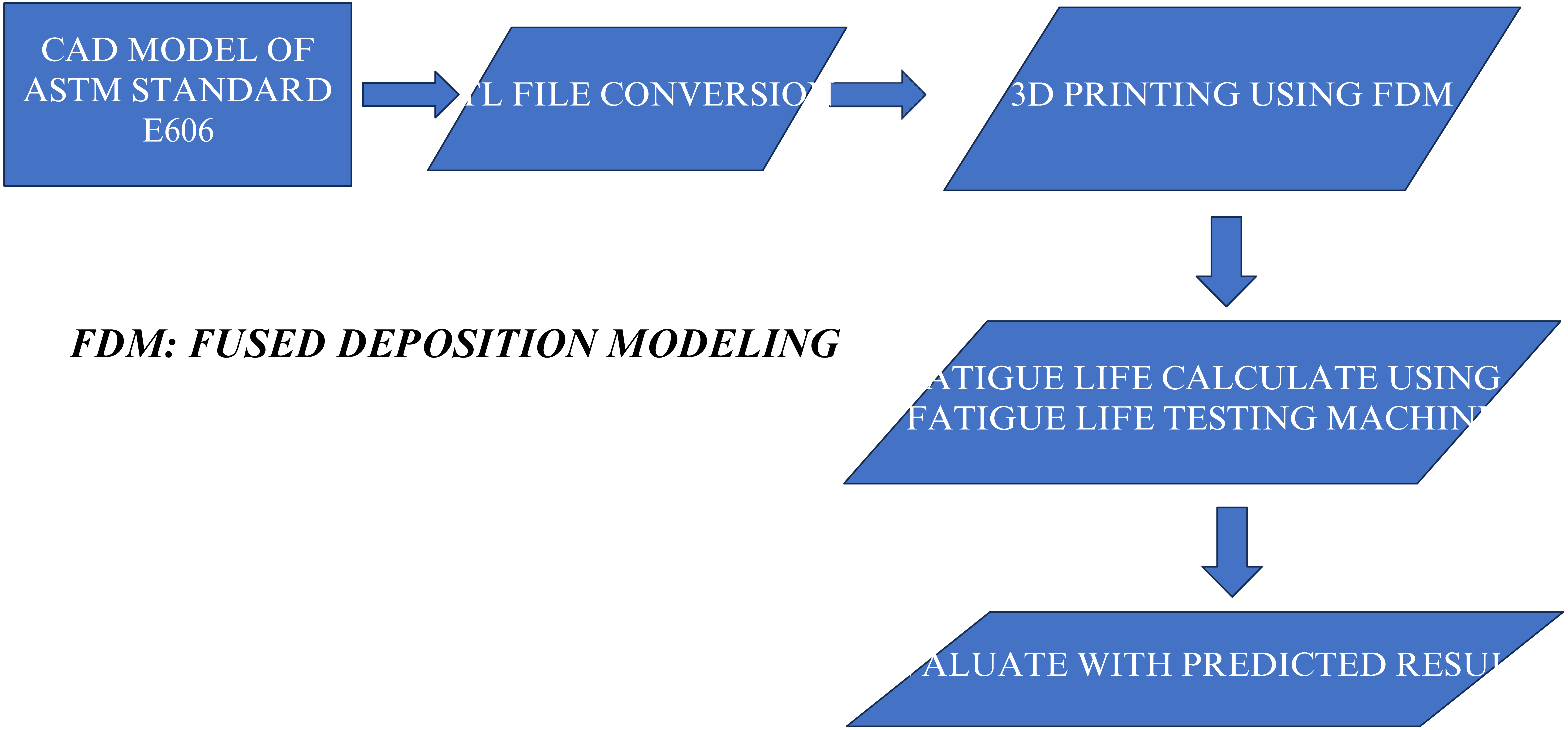


Iteration Values Table:



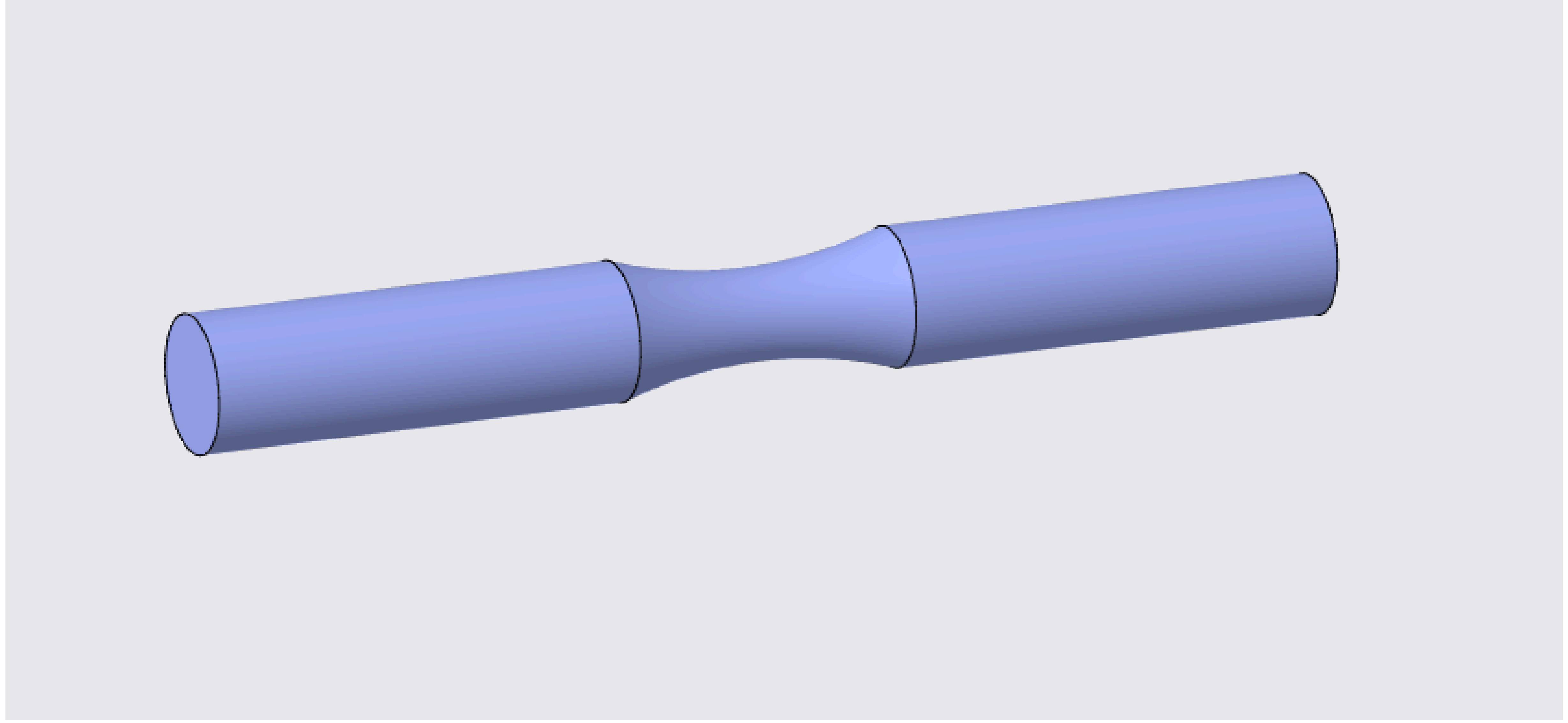
Iteration	Fatigue Lifetime	Nozzle Temperature	Nozzle Diameter	Print Speed	Stress Level
1	1468280	197.8066222	0.222668013	9.98198368	3.20691827
2	1468280	197.8066222	0.222668013	9.98198368	3.20691827
3	1468280	197.8066222	0.222668013	9.98198368	3.20691827
4	1468280	197.8066222	0.222668013	9.98198368	3.20691827
5	1479440	215.5823087	0.2282233	6.730927518	3.447202733
6	1485020	224.7108171	0.215529301	8.875059536	3.153144303
7	1485020	224.7108171	0.215529301	8.875059536	3.153144303
8	1485020	224.7108171	0.215529301	8.875059536	3.153144303
9	1485020	225.2529422	0.215529301	8.360292198	3.737605097
10	1485020	225.2529422	0.215529301	8.360292198	3.737605097
11	1485020	224.0519546	0.262123318	8.630714281	3.737605097
12	1485020	224.0519546	0.262123318	8.630714281	3.737605097
13	1485020	224.0519546	0.262123318	8.630714281	3.737605097

EVALUATION METHODOLOGY:

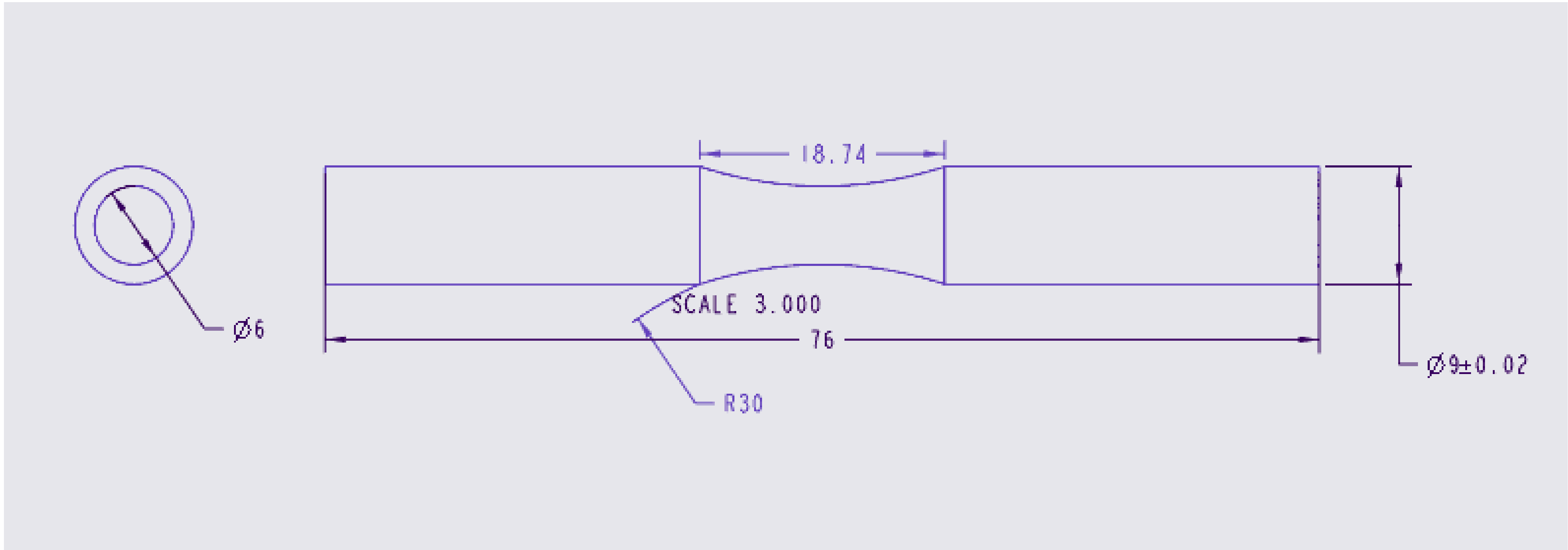

CAD MODEL OF ASTM STANDARD E606
TL FILE CONVERSIO
3D PRINTING USING FDM
ATIGUE LIFE CALCULATE USING FATIGUE LIFE TESTING MACHIN
ALUATE WITH PREDICTED RESUL

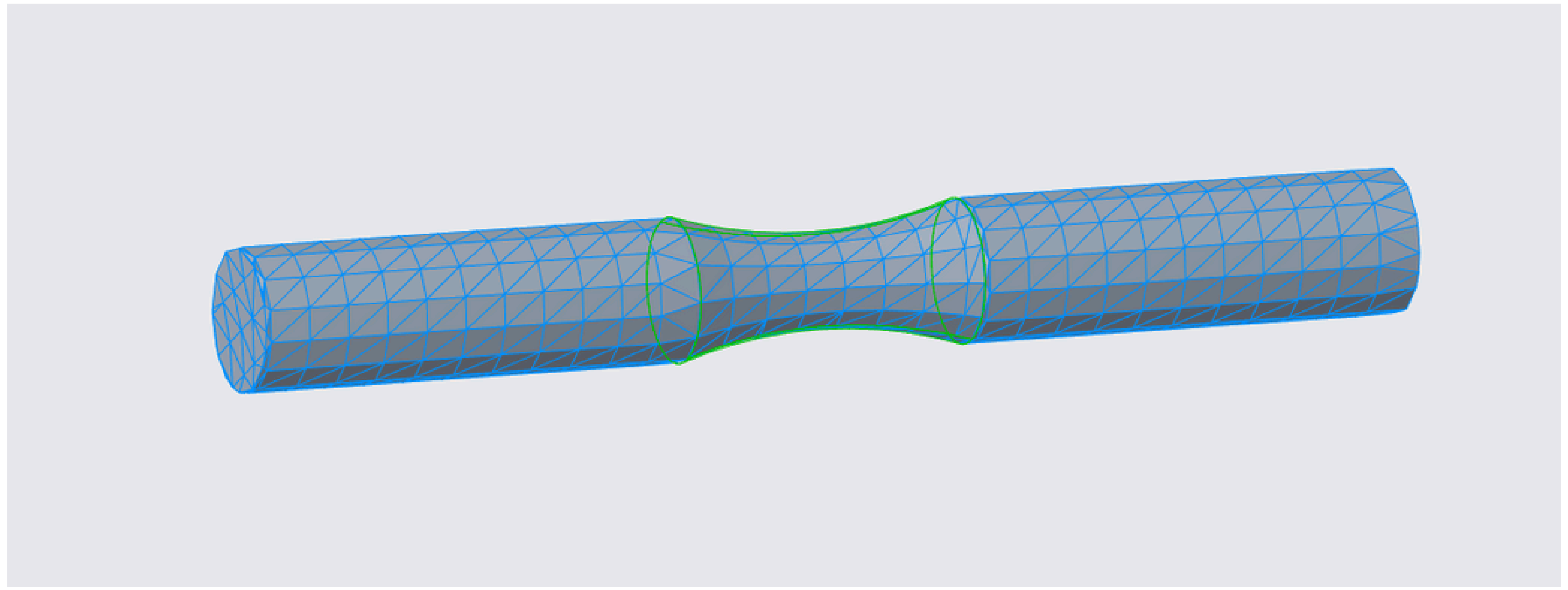
FDM: FUSED DEPOSITION MODELING

Fatigue testing standard [Creo parametric]:



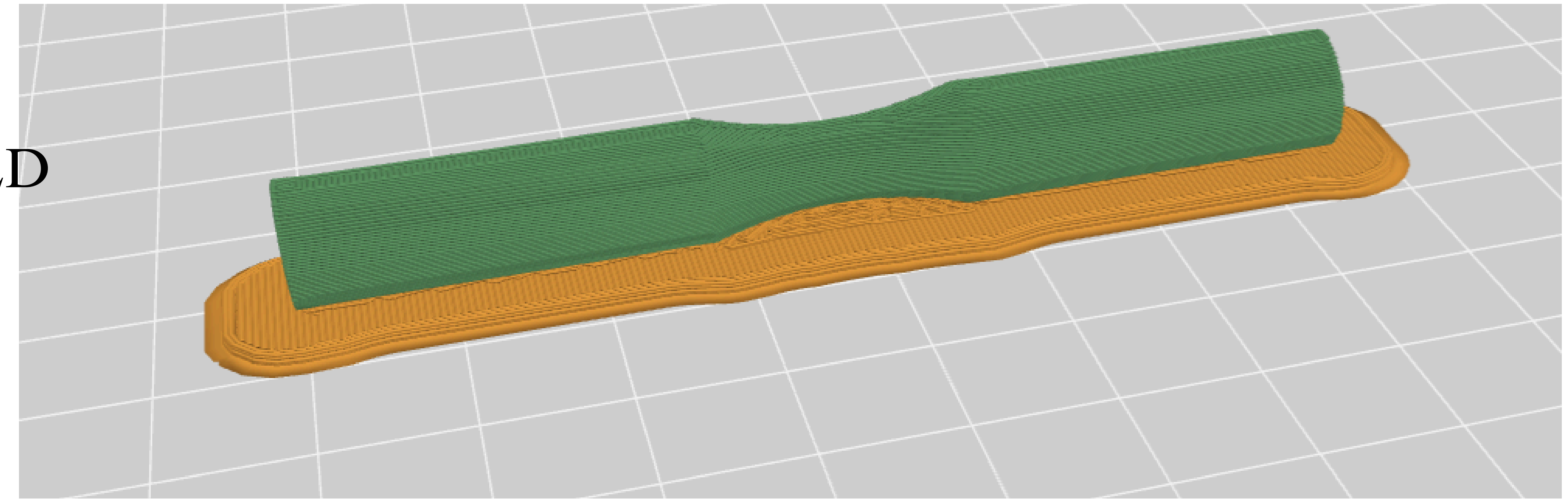
Cad drawing:





STL FILE IMAGE with 944 Triangles

MAKER BOT PRINT SIMULATED IMAGE



PLA material consumption = 8.28g with print speed = 10mm/s, Extruder Temp = 235℃, infill density 95% and support structure.
Estimate time to print = 1hr 16 mins

References:

- Experimental fatigue dataset for additive-manufactured 3D-printed Polylactic acid biomaterials under fully-reversed rotating-bending bending loadings Mohammad Azadi , Ali Dadashi (2022)
- Data-driven Approach to Predict the Static and Fatigue Properties of Additively Manufactured Ti-6Al-4V Antriksh Sharma (2020)
- Characterization and parametric optimization of additive manufacturing process for enhancing mechanical properties   Amanuel Diriba Tura , Hana Beyene Mamo (2022)
- Single and Multi-Objective Optimisation of Processing Parameters for Fused Deposition Modelling in 3D Printing Technology V.H. Nguyen, T.N. Huynh, T.P. Nguyen and T.T. Tran (2020)
- A Parametric study of 3D printed polymer gears Ye Zhang , Ken Mao , Simon Leigh , Guotao Ma (2020)
- Modelling and parametric optimization of FDM 3D printing process using hybrid techniques for enhancing dimensional preciseness   Sandeep Deswal , Rajan Narang , Deepak Chhabra (2019)

Work yet to be carried out:

- Validation of predicted and optimized fatigue life cycle with fatigue test

CONCLUSION:

ML MODELS	Nozzle diameter [mm]	Nozzle temperature [℃]	Stress level [MPa]	Print speed [mm/s]	Fatigue Life Predicted value [Cycles]	Validation Fatigue Life [Cycles]
Linear regression	0.27	235	2.6	9.2	392325.15	
Decision tree	0.27	235	2.6	9.2	1493328.97	
Random Forest	0.27	235	2.6	9.2	1501167.99	
Gradient boost	0.27	235	2.6	9.2	1579765.78	
Xgboost	0.27	235	2.6	9.2	1507820.12	
Differential Evolution	0.26	224.05	3.74	8.63	1485020.00	