```
import pandas as pd
import random
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
from deap import base, creator, tools, algorithms

df = pd.read_excel("CNCDATASETCLEANED.xlsx")

X = df.drop("Tool_Wear_mm", axis=1)
y = df["Tool_Wear_mm"]

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

n_feat = X_train.shape[1]
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, n=n_feat)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

def eval(ind):
    idx = [i for i, b in enumerate(ind) if b == 1]
    if not idx: return 9999.0,
    model = DecisionTreeRegressor()
    model.fit(X_train.iloc[:, idx], y_train)
    pred = model.predict(X_test.iloc[:, idx])
    return mean_squared_error(y_test, pred),

toolbox.register("evaluate", eval)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.1)
toolbox.register("select", tools.selTournament, tournsize=3)

pop = toolbox.population(n=20)
for g in range(15):
    offspring = algorithms.varAnd(pop, toolbox, cxpb=0.5, mutpb=0.2)
    fits = toolbox.map(toolbox.evaluate, offspring)
    for fit, ind in zip(fits, offspring): ind.fitness.values = fit
    pop = toolbox.select(offspring, k=len(pop))

best = tools.selBest(pop, k=1)[0]
idx = [i for i, b in enumerate(best) if b == 1]
print("features:", list(X.columns[idx]))

model = DecisionTreeRegressor()
model.fit(X_train.iloc[:, idx], y_train)
pred = model.predict(X_test.iloc[:, idx])

print("mse:", mean_squared_error(y_test, pred))
print("r2:", r2_score(y_test, pred))
```

```
features: ['Spindle_Speed_RPM', 'Feed_Rate_mm_per_min', 'Depth_of_Cut_mm', 'Cutting_Time_min', 'Tool_Temperature_
mse: 0.9109443315648292
r2: 0.6063817448207035
```

```
pip install deap
```

```
Collecting deap
  Downloading deap-1.4.3-cp311-cp311-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from deap) (2.0.2)
Downloading deap-1.4.3-cp311-cp311-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.6/135.6 kB 5.6 MB/s eta 0:00:00
Installing collected packages: deap
Successfully installed deap-1.4.3
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

df = pd.read_excel("CNCDATASETCLEANED.xlsx")

X = df.drop(columns='Tool_Wear_mm')
y = df['Tool_Wear_mm']

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

models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(random_state=42)
}


for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    print(f"✅ {name} R² Score: {r2:.3f}")


best_model = DecisionTreeRegressor(random_state=42)
best_model.fit(X_train, y_train)
y_pred = best_model.predict(X_test)

plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.6, color='green')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual Tool Wear (mm)")
plt.ylabel("Predicted Tool Wear (mm)")
plt.title("Decision Tree: Actual vs Predicted Tool Wear")
plt.grid(True)
plt.show()
```

Linear Regression R² Score: 0.891
Decision Tree R² Score: 0.605