```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Load the data
df = pd.read_csv("processed_features.csv")

# Define features (X) and target (y)
target = 'Mu'
# Exclude the target, and categorical/ID columns for this initial model
features_to_exclude = ['Mu', 'Solute_encoded', 'Solvent_encoded', 'Potentia
X = df.drop(columns=features_to_exclude)
y = df[target]

# Split the data into training and testing sets (90% train, 10% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, ra

# 1. Decision Tree Regression
# Model initialization
dt_regressor = DecisionTreeRegressor(random_state=42)
# Training the model
dt_regressor.fit(X_train, y_train)
# Predictions on test set
y_pred_dt = dt_regressor.predict(X_test)
# Evaluation metrics
dt_mse = mean_squared_error(y_test, y_pred_dt)
dt_rmse = np.sqrt(dt_mse)
dt_r2 = r2_score(y_test, y_pred_dt)

# 2. Random Forest Regression
# Model initialization (using 2000 trees)
rf_regressor = RandomForestRegressor(n_estimators=2000, random_state=42, n_
# Training the model
rf_regressor.fit(X_train, y_train)
# Predictions on test set
y_pred_rf = rf_regressor.predict(X_test)
# Evaluation metrics
rf_mse = mean_squared_error(y_test, y_pred_rf)
rf_rmse = np.sqrt(rf_mse)
rf_r2 = r2_score(y_test, y_pred_rf)

# Print results
print("Decision Tree RMSE:", dt_rmse)
print("Decision Tree R2:", dt_r2)
print("Random Forest RMSE:", rf_rmse)
print("Random Forest R2:", rf_r2)
```

```
Decision Tree RMSE: 11.477987020663145
Decision Tree R2: -0.07117101270832382
Random Forest RMSE: 9.912501220045316
```

```
Random Forest R2: 0.20109735223274106
```

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint, uniform

# Define hyperparameter distributions instead of fixed lists
param_dist = {
    'n_estimators': randint(100,3000),  # you could also use randint(100,100
    'max_depth': [None, 10, 20, 30, 40],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4, 8],
    'max_features': ['sqrt', 'log2', None]
}

rf = RandomForestRegressor(random_state=42, n_jobs=-1)

random_search = RandomizedSearchCV(
    estimator=rf,
    param_distributions=param_dist,
    n_iter=50,             # number of random combinations to try
    cv=5,
    scoring='r2',
    random_state=42,
    n_jobs=-1,
    verbose=2
)

random_search.fit(X_train, y_train)

best_model = random_search.best_estimator_
y_pred_best = best_model.predict(X_test)

print("Best Parameters:", random_search.best_params_)
print("Best R2:", r2_score(y_test, y_pred_best))
import numpy as np
best_mse = mean_squared_error(y_test, y_pred_best)
best_rmse = np.sqrt(best_mse)
print("Best RMSE:", best_rmse)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
Best Parameters: {'max_depth': 20, 'max_features': None, 'min_samples_leaf':
Best R2: 0.32227725891396153
Best RMSE: 9.129822893295357
```