```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

# Gerar dados simulados
np.random.seed(42)
corrente_a_carregar = np.random.uniform(300, 500, 500)
corrente_carregado = np.random.uniform(0, 50, 500)

# Etiquetas: 0 para carregando, 1 para carregado
y = np.array([0]*500 + [1]*500)
X = np.concatenate([corrente_a_carregar, corrente_carregado]).reshape(-1, 1)

# Dividir dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalizar os dados
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Criar a rede neural com arquitetura 1-3-2-1
modelo = MLPClassifier(hidden_layer_sizes=(3, 2), max_iter=1000, random_state=42)
```

```
# Treinar o modelo
modelo.fit(X_train, y_train)

# Avaliar o modelo
precisao = modelo.score(X_test, y_test)
print(f"Precisão no teste: {precisao:.2f}")

# Visualizar a fronteira de decisão
x_vals = np.linspace(0, 500, 1000).reshape(-1, 1)
x_vals_scaled = scaler.transform(x_vals)
y_pred = modelo.predict(x_vals_scaled)

plt.plot(x_vals, y_pred, color='red', label='Decisão do modelo')
plt.legend()
plt.title("Fronteira de Decisão - Rede 1-3-2-1")
plt.xlabel("Corrente (mA)")
plt.ylabel("Classe")
plt.grid(True)
plt.show()
```