```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import xgboost as xgb
import matplotlib.pyplot as plt

# Підготовка даних
data = pd.DataFrame({
    'Slice_Type': ['eMBB', 'URLLC', 'mMTC'],
    'Min_CPU': [20, 15, 10],
    'Min_RAM': [10, 10, 5],
    'Min_BW': [30, 20, 10],
    'Load': [70, 50, 30],
    'Optimal_CPU': [30, 20, 12],
    'Optimal_RAM': [15, 12, 8],
    'Optimal_BW': [40, 25, 15]
})

# One-hot encoding для типу слайсу
data = pd.get_dummies(data, columns=['Slice_Type'])

# Вхідні та цільові змінні
X = data.drop(columns=['Optimal_CPU', 'Optimal_RAM', 'Optimal_BW'])
y = data[['Optimal_CPU', 'Optimal_RAM', 'Optimal_BW']]

# Розподіл даних на тренувальні та тестові
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
        # Навчання моделі для CPU
        model_cpu = xgb.XGBRegressor(objective ='reg:squarederror',
eval_metric='rmse')
        model_cpu.fit(X_train, y_train['Optimal_CPU'])

        # Передбачення для CPU
        pred_cpu = model_cpu.predict(X_test)

        # Оцінка моделі для CPU
        mse_cpu = mean_squared_error(y_test['Optimal_CPU'], pred_cpu)
        rmse_cpu = np.sqrt(mse_cpu)
        print(f'MSE for CPU: {mse_cpu}')
        print(f'RMSE for CPU: {rmse_cpu}')

        # Навчання моделі для RAM
        model_ram = xgb.XGBRegressor(objective ='reg:squarederror',
eval_metric='rmse')
        model_ram.fit(X_train, y_train['Optimal_RAM'])

        # Передбачення для RAM
        pred_ram = model_ram.predict(X_test)

        # Оцінка моделі для RAM
        mse_ram = mean_squared_error(y_test['Optimal_RAM'], pred_ram)
        rmse_ram = np.sqrt(mse_ram)
        print(f'MSE for RAM: {mse_ram}')
        print(f'RMSE for RAM: {rmse_ram}')

        # Навчання моделі для BW
```

```
model_bw = xgb.XGBRegressor(objective ='reg:squarederror',
eval_metric='rmse')
model_bw.fit(X_train, y_train['Optimal_BW'])

# Передбачення для BW
pred_bw = model_bw.predict(X_test)

# Оцінка моделі для BW
mse_bw = mean_squared_error(y_test['Optimal_BW'], pred_bw)
rmse_bw = np.sqrt(mse_bw)
print(f'MSE for BW: {mse_bw}')
print(f'RMSE for BW: {rmse_bw}')

# Візуалізація результатів для передбачених та фактичних значень
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
plt.scatter(y_test['Optimal_CPU'], pred_cpu)
plt.title('CPU: Actual vs Predicted')
plt.xlabel('Actual CPU')
plt.ylabel('Predicted CPU')

plt.subplot(1, 3, 2)
plt.scatter(y_test['Optimal_RAM'], pred_ram)
plt.title('RAM: Actual vs Predicted')
plt.xlabel('Actual RAM')
plt.ylabel('Predicted RAM')

plt.subplot(1, 3, 3)
plt.scatter(y_test['Optimal_BW'], pred_bw)
```

```
plt.title('BW: Actual vs Predicted')
plt.xlabel('Actual BW')
plt.ylabel('Predicted BW')

plt.tight_layout()
plt.show()

# Підсумкові результати для оптимального розподілу ресурсів
print("\nOptimized resource distribution:")

for idx, row in X_test.iterrows():
    print(f"Slice Type: {row['Slice_Type_eMBB']*'eMBB' +
row['Slice_Type_URLLC']*'URLLC' + row['Slice_Type_mMTC']*'mMTC'}")
    print(f"Optimal CPU: {model_cpu.predict([row])[0]}")
    print(f"Optimal RAM: {model_ram.predict([row])[0]}")
    print(f"Optimal BW: {model_bw.predict([row])[0]}")
    print('-' * 40)
```

Врахування важливості ознак для кожного слайсу та використання їх для подальшого вдосконалення моделі.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
import xgboost as xgb
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Підготовка даних
data = pd.DataFrame({
```

```
        'Slice_Type': ['eMBB', 'URLLC', 'mMTC'],
        'Min_CPU': [20, 15, 10],
        'Min_RAM': [10, 10, 5],
        'Min_BW': [30, 20, 10],
        'Load': [70, 50, 30],
        'Optimal_CPU': [30, 20, 12],
        'Optimal_RAM': [15, 12, 8],
        'Optimal_BW': [40, 25, 15]
    })

    # One-hot encoding для типу слайсу
    data = pd.get_dummies(data, columns=['Slice_Type'])

    # Вхідні та цільові змінні
    X = data.drop(columns=['Optimal_CPU', 'Optimal_RAM', 'Optimal_BW'])
    y = data[['Optimal_CPU', 'Optimal_RAM', 'Optimal_BW']]

    # Нормалізація даних
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # Розподіл даних на тренувальні та тестові
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

    # Створення і налаштування моделі XGBoost
    model_cpu = xgb.XGBRegressor(objective='reg:squarederror',
eval_metric='rmse',
                                 max_depth=5, learning_rate=0.1, n_estimators=100,
subsample=0.8)
```

```
        # Навчання моделі для CPU
        model_cpu.fit(X_train, y_train['Optimal_CPU'])

        # Оцінка моделі для CPU
        pred_cpu = model_cpu.predict(X_test)
        mse_cpu = mean_squared_error(y_test['Optimal_CPU'], pred_cpu)
        rmse_cpu = np.sqrt(mse_cpu)
        print(f'MSE for CPU: {mse_cpu}')
        print(f'RMSE for CPU: {rmse_cpu}')

        # Навчання моделі для RAM
        model_ram = xgb.XGBRegressor(objective='reg:squarederror',
eval_metric='rmse',
                                     max_depth=5, learning_rate=0.1, n_estimators=100,
subsample=0.8)

        # Навчання моделі для RAM
        model_ram.fit(X_train, y_train['Optimal_RAM'])

        # Оцінка моделі для RAM
        pred_ram = model_ram.predict(X_test)
        mse_ram = mean_squared_error(y_test['Optimal_RAM'], pred_ram)
        rmse_ram = np.sqrt(mse_ram)
        print(f'MSE for RAM: {mse_ram}')
        print(f'RMSE for RAM: {rmse_ram}')

        # Навчання моделі для BW
        model_bw = xgb.XGBRegressor(objective='reg:squarederror',
eval_metric='rmse',
```

```
                    max_depth=5, learning_rate=0.1, n_estimators=100,
subsample=0.8)

    # Навчання моделі для BW
    model_bw.fit(X_train, y_train['Optimal_BW'])

    # Оцінка моделі для BW
    pred_bw = model_bw.predict(X_test)
    mse_bw = mean_squared_error(y_test['Optimal_BW'], pred_bw)
    rmse_bw = np.sqrt(mse_bw)
    print(f'MSE for BW: {mse_bw}')
    print(f'RMSE for BW: {rmse_bw}')

    # Візуалізація результатів
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 3, 1)
    plt.scatter(y_test['Optimal_CPU'], pred_cpu)
    plt.title('CPU: Actual vs Predicted')
    plt.xlabel('Actual CPU')
    plt.ylabel('Predicted CPU')

    plt.subplot(1, 3, 2)
    plt.scatter(y_test['Optimal_RAM'], pred_ram)
    plt.title('RAM: Actual vs Predicted')
    plt.xlabel('Actual RAM')
    plt.ylabel('Predicted RAM')

    plt.subplot(1, 3, 3)
    plt.scatter(y_test['Optimal_BW'], pred_bw)
```

```
        plt.title('BW: Actual vs Predicted')
        plt.xlabel('Actual BW')
        plt.ylabel('Predicted BW')

        plt.tight_layout()
        plt.show()

        # Підсумкові результати для оптимального розподілу ресурсів
        print("\nOptimized resource distribution:")

        for idx, row in X_test.iterrows():
            print(f"Slice Type: {row[0]}")  # Виводимо тип слайсу
(експериментальна реалізація)
            print(f"Optimal CPU: {model_cpu.predict([row])[0]}")
            print(f"Optimal RAM: {model_ram.predict([row])[0]}")
            print(f"Optimal BW: {model_bw.predict([row])[0]}")
            print('-' * 40)
```