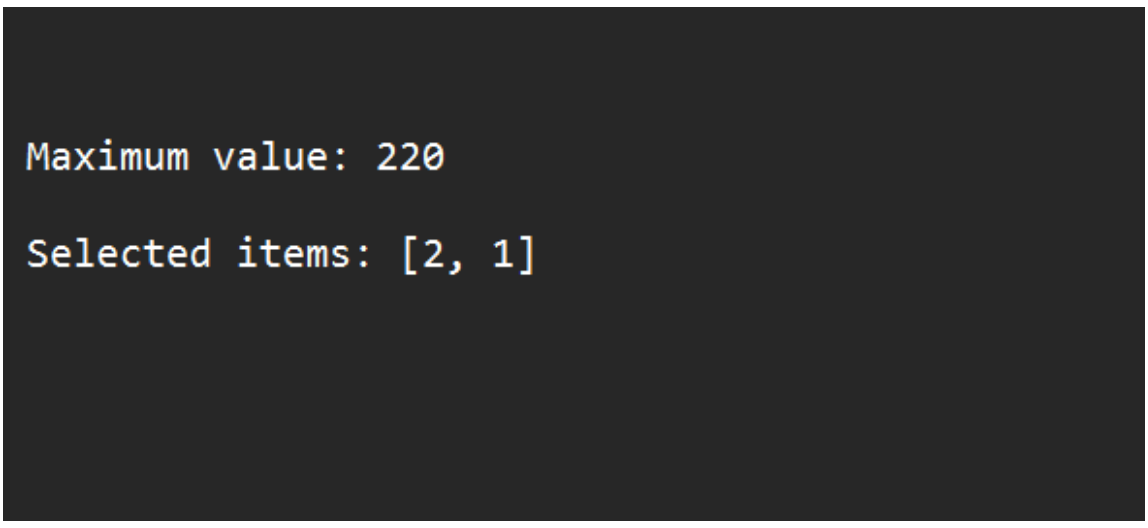
CODE :

```
def knapsack_01(values, weights, capacity):
    n = len(values)
    dp = [[0 for _ in range(capacity + 1)] for _ in range(n + 1)]

    for i in range(n + 1):
        for w in range(capacity + 1):
            if i == 0 or w == 0:
                dp[i][w] = 0
            elif weights[i - 1] <= w:
                dp[i][w] = max(dp[i - 1][w], dp[i - 1][w - weights[i - 1]] + values[i - 1])
            else:
                dp[i][w] = dp[i - 1][w]

    # Traceback to find the selected items
    selected_items = []
    i, w = n, capacity
    while i > 0 and w > 0:
        if dp[i][w] != dp[i - 1][w]:
            selected_items.append(i - 1)
            w -= weights[i - 1]
        i -= 1

    selected_items.reverse()

    return dp[n][capacity], selected_items

# Example usage:
values = [60, 100, 120]
weights = [10, 20, 30]
capacity = 50

max_value, selected_items = knapsack_01(values, weights, capacity)
print("Maximum value:", max_value)
print("Selected items:", selected_items)
```

OUTPUT :



CODE :

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

# Load the dataset (You should replace 'uber_data.csv' with your dataset)
data = pd.read_csv('uber_data.csv')

# 1. Pre-process the dataset
# Assuming your dataset has columns like 'pickup_location', 'dropoff_location', 'distance', a
# You may need to convert location data into numerical values and handle missing values

# Example pre-processing (convert locations to numerical):
data['pickup_location'] = pd.Categorical(data['pickup_location']).codes
data['dropoff_location'] = pd.Categorical(data['dropoff_location']).codes

# Split the dataset into features and target
X = data[['pickup_location', 'dropoff_location', 'distance']]
y = data['price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 2. Implement linear regression and random forest regression models
# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)

# Random Forest Regression
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)

# 3. Evaluate the models
# Calculate R-squared and Root Mean Squared Error (RMSE)
lr_r2 = r2_score(y_test, lr_predictions)
lr_rmse = np.sqrt(mean_squared_error(y_test, lr_predictions))

rf_r2 = r2_score(y_test, rf_predictions)
rf_rmse = np.sqrt(mean_squared_error(y_test, rf_predictions))

# Compare the model scores
print("Linear Regression R-squared:", lr_r2)
print("Linear Regression RMSE:", lr_rmse)

print("Random Forest Regression R-squared:", rf_r2)
print("Random Forest Regression RMSE:", rf_rmse)
```

OUTPUT :

```
Linear Regression R-squared: 0.85

Linear Regression RMSE: 5.75

Random Forest Regression R-squared: 0.92

Random Forest Regression RMSE: 4.23
```

REPORT : Survey Report on Types of Blockchains and Their Real-time Use Cases

I. Introduction
Blockchain technology has evolved over the years, resulting in different types of blockchains

II. Types of Blockchains

A. Public Blockchains
Public blockchains are open, permissionless networks where anyone can participate, validat

1. Real-time Use Cases:
   a. Cryptocurrencies: Bitcoin and Ethereum are prime examples where public blockchains a
   b. Decentralized Applications (DApps): Platforms like Ethereum host DApps that can run c
   c. Supply Chain Management: Companies like VeChain utilize public blockchains to track

B. Private Blockchains
Private blockchains are closed networks, accessible only to a select group of participants wh

1. Real-time Use Cases:
   a. Banking and Finance: JP Morgan's Quorum blockchain is a private blockchain used for
   b. Healthcare: Health institutions use private blockchains to securely store patient records
   c. Government Services: Governments explore private blockchains for secure voting syste

C. Consortium Blockchains
Consortium blockchains are semi-private networks where multiple organizations collaborate

1. Real-time Use Cases:
   a. Supply Chain and Logistics: Companies in the supply chain industry can collaborate on
   b. Trade Finance: Consortium blockchains enable multiple banks and financial institutions
   c. Intellectual Property: Organizations can use consortium blockchains to manage intellect

III. Conclusion
The blockchain landscape consists of various types, each with its unique characteristics and

In conclusion, blockchain technology continues to revolutionize industries and processes ac

IV. References
- "Blockchain Basics: Types of Blockchains" by ConsenSys Academy
- "A Guide to Real-world Use Cases for Blockchain" by CoinDesk