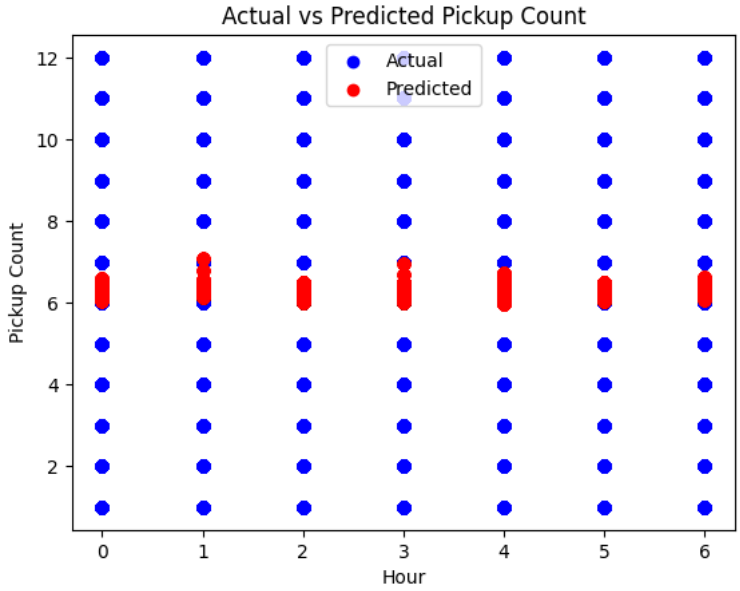
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Load and preprocess the data
data = pd.read_csv('uber.csv')
data['pickup_datetime'] = pd.to_datetime(data['pickup_datetime'])  # Convert to datetime object
data['hour'] = data['pickup_datetime'].dt.hour  # Extract hour of the day
data['month'] = data['pickup_datetime'].dt.month
data['day'] = data['pickup_datetime'].dt.day
data['day_of_week'] = data['pickup_datetime'].dt.dayofweek  # Extract day of the week
X = data[['hour', 'day_of_week']].values  # Input features as numerical values
y = data['day'].values  # Target variable as numerical values

# 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)

# Create and train the ML model
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Evaluate the model
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("Mean Squared Error:", mse)

# Make predictions for new data
new_data = pd.DataFrame({'hour': [8, 12, 18], 'day_of_week': [1, 3, 6]})  # Example new data
new_predictions = model.predict(new_data)

# Perform further analysis and decision-making based on the predictions
```

```
Mean Squared Error: 11.849642426559724
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but RandomForestRegressor was fitted
  warnings.warn(
```

```
# Check the shape of X_test and confirm the column index for the hour column
print("X_test shape:", X_test.shape)
```

```
X_test shape: (40000, 2)
```

```
# Continue from previous code

# Perform further analysis and decision-making based on the predictions

# Visualize actual vs predicted values
import matplotlib.pyplot as plt

plt.scatter(X_test[:, 1], y_test, color='blue', label='Actual')  # Access the hour column using index 1
plt.scatter(X_test[:, 1], predictions, color='red', label='Predicted')  # Access the hour column using index 1
plt.xlabel('Hour')
plt.ylabel('Pickup Count')
plt.title('Actual vs Predicted Pickup Count')
plt.legend()
plt.show()

# Identify peak hours
peak_hours = pd.Series(X_test[:, 1]).value_counts().sort_values(ascending=False).head(3)  # Access the hour column using index 1
print("Peak Hours:", peak_hours)

# Make decisions based on predictions
threshold = 100  # Example threshold value
if new_predictions[0] > threshold:
    print("Expected high demand during the specified hour.")
    # Take appropriate actions, such as allocating more drivers or offering promotions.
else:
    print("Expected normal demand during the specified hour.")
    # Continue with regular operations.
```




```
Peak Hours: 4    6216
5    6124
3    6009
dtype: int64
Expected normal demand during the specified hour.
```