These are the steps for building a regression model for the demand-supply gap prediction problem:

Data Collection: Collect data on the demand and supply of rides in each region and time slot, along with other relevant features such as weather, events, and time of day. This data can be collected using Uber's app and other sources.

Data Preprocessing: Clean the data and handle missing values, outliers, and other anomalies. Convert categorical variables to numerical using techniques such as one-hot encoding or label encoding. Split the data into training, validation, and test sets.

Feature Engineering: Create new features from the existing ones that might improve the model's performance. For example, create binary variables to indicate whether it's a weekday or weekend, or create lag features to capture the trends in the demand and supply over time.

Model Selection: Choose a regression algorithm that is appropriate for the problem, such as linear regression, decision trees, or random forests. Tune the hyperparameters of the model using techniques such as grid search or random search.

Model Training: Train the model on the training set and evaluate its performance on the validation set. Use techniques such as cross-validation and regularization to avoid overfitting and improve the model's generalization performance.

Model Evaluation: Evaluate the model's performance on the test set using metrics such as mean absolute error, mean squared error, and R-squared. Compare the model's performance with baseline models such as mean or median prediction.

Model Deployment: Deploy the model in a production environment, where it can make predictions in real-time based on new data. Monitor the model's performance regularly and make updates and improvements as needed.

**Step1**

I downloaded the data from the google classroom.

**Step2**

I wrote python code that reads various data files from a folder called 'training\_data' and merges them together using pandas.

First, it reads in a file called 'cluster\_map' and creates a dictionary called 'cluster\_map' from its contents.

Then, it reads in a file called 'order\_data' and assigns it to the variable 'order\_data'. The 'header=None' argument tells pandas that there is no header row in the file, and the 'names' argument provides the column names.

Next, it reads in a file called 'poi\_data' and assigns it to the variable 'poi\_data', using the same arguments as for 'order\_data'.

It then reads in a file called 'weather\_data' and assigns it to the variable 'weather\_data', using the same arguments as for 'order\_data'.

Finally, it reads in all files ending with '.csv' from a folder called 'traffic\_data' and merges them together into a single dataframe called 'traffic\_data', adding a 'region\_hash' column to each file's data before concatenating.

The code then merges the 'order\_data' dataframe with the 'cluster\_map' dictionary to add region names to the 'start\_region' and 'dest\_region' columns. Similarly, it merges the 'poi\_data' dataframe with the 'cluster\_map' dictionary to add region names to the 'region' column.

**Step 3**

Then I wrote the code to preprocess the training data by reading in the necessary files and merging them together. It then drops some unnecessary columns and encodes categorical variables using LabelEncoder. Finally, it fills in missing values with 0 and writes the preprocessed data to a file called data.csv.

**Step 4**

Then we implemented a decision tree algorithm for a machine learning task. The algorithm is using entropy and information gain to determine the best features to split the data on and recursively building a tree until a stopping criterion is met. The code also includes functions to split the data into training and testing sets, make predictions using the decision tree, and calculate accuracy.

The code loads preprocessed data from a CSV file, defines the target variable and the features to use for prediction, splits the data into training and testing sets, builds the decision tree on the training data, makes predictions on the test data, and calculates the accuracy of the predictions.