Problem Statement

Our machine learning system aims to address the problem of shipment time, i.e. a system which after receiving user's order request, could basically predict the time interval in which the user could receive their order. This machine learning system would be useful for the customer to get an estimate of the order's processing time and also help the companies in improving their respective metrics around product delivery thereby driving value.

Data Collection and Preprocessing

The following data sources could be used to build the dataset for training and evaluation of the model

1) User Database – Which will store information like user's delivery address (zip code)
2) Product Database – Which will have information like product id, weight and dimensions of the product
3) Warehouse database – Which will store information of the warehouse which containing that product
4) Product Order database – Which for every product ID will store information like date of order, date of actual delivery, was it a bulk order, total shipment days, shipping priority

The data preprocessing step would look for any missing values or outliers in these data sources and would process the dataset to have a curated dataset.

Feature Engineering:

The following features could be used for building the model:
1) Distance between user and shipper
2) Order's weight
3) Order's width
4) Order's height
5) Order's breadth
6) Date of Order
7) Date of Actual Delivery
8) Was Bulk Order
9) Carrier Used
10) Shipping Priority
11) Date of Order Placement
12) Holiday

NOTE
Here Date of Order Placement can be used along with Date of Actual Delivery to derive the total time it took to deliver the order, which is the response variable. Next, Date of Order Placement can be used to extract day and month of order placement.

The following approach could be taken to do feature selection:
a) Pearson's Correlation
b) ANOVA (Analysis of Variance @ defined confidence interval)
These feature selection tests would basically help us in preventing overfitting and help reduce the overall error.

Model Selection and Evaluation

We are trying to frame the problem as a regression problem and thus to set up a baseline we can use a Linear Regression problem. The model would be evaluated on RMSE score values and thus after setting up a baseline on the test set, we can then choose a tree-based model particularly XGBoost to compare with the baseline and decide. In case XGBoost performs better than Linear Regression, then feature importance scores can be used to evaluate the importance of each feature, which can then server as an additional criterion to evaluate the features and do a final selection of the feature set. The chosen features can then be used to retrain the model which would be lighter and easy to deploy.

For data points where the time it took to deliver the order was high, because say the order was placed from distant locations like Alaska or Hawaii, we need to ensure that we include more such data point as part of the training set so that the model is able to generalize well and the overall RMSE score is lower.

Training and Optimization

The dataset collected would then be divided into Training and Testing set, The training set could further be split into training and validation datasets. After comparing the Linear Regression Model's performance with XGBoost Model, if we select XGBoostRegressor model we can train the model on the training set and tune the parameters like number of trees, column sampling, row sampling, learning rate etc to finetune out model. An optimal way to try out multiple parameter values would be to use RandomSearch. Additionally, to prevent overfitting we can use L1 and L2 regularization. All of this can be wrapped inside a pipeline object which after being trained can give us the best combination of hyperparameter values and the best estimator.

Deployment and Monitoring:

The model object could be packaged as a docker image and then deployed on AWS. The serve script could also use gunicorn server to scale the inference requests. The deployed endpoint could be set to Auto Scaling so as to handle multiple inference requests every time different users places their orders. AWS Model monitor could be used to capture the data and concept drift which can then be compared against the test set's performance trigger a model retraining event.

Ethical Considerations

None

Reflection and Future Improvements: