1. Provide a high level summary of your approach: what did you do and why?
Our overall approach was to adapt the method for horizontally federating XGBoost developed by Ma et al. (paper, code). We adjusted the code in order to make it work with CatBoost, which was the architecture we used in Phase 1. We also made some adjustments to speed up inference.

2. What ways did you consider adapting your Phase 1 approach to accommodate federated learning? Which did you eventually implement and why?
We initially considered implementing strategies such as Federated Averaging and Adaptive Federated Optimization to aggregate the CatBoost models trained by clients. However, we settled on using Ma et al.'s "FedXgbNnAvg" strategy (Federated XGBoost Neural Network Averaging) because we felt it had the most appropriate level of complexity for the task. In brief, our implementation works as follows:

1. First, each airline trains a CatBoost model with 50 trees on their respective dataset.
2. Next, all the trees trained by all of the airlines are aggregated and arranged into a fixed order by the central server. The aggregated list of trees is then shared with each airline.
3. Next, each airline trains a CNN that takes the predictions of each tree as input and outputs a final prediction for the pushback times.
4. The central server averages the weights of the trained CNNs.
5. The resultant CNN is sent back to the airlines to continue training. Steps 3-4 are repeated for a total of 10 rounds.

3. Compare the performance (in terms of mean absolute error or other relevant metrics) of your Phase 1 and Phase 2 models.
Overall, our Phase 2 model returned scores that were about 30% worse on the test set. MAE's in Phase 2 varied between 10.9 and 20.5 depending on the airport.

4. What experiments did you undertake to optimize performance? Do you believe your final Phase 2 submission achieves the best possible performance of a federated version of your Phase 1 model? If not, what do you think would be needed to achieve the best possible performance? What are the trade-offs involved between performance and privacy for your Phase 2 solution?
Given the large amount of data we were provided, we implemented two changes to the FedXgbNnAvg strategy to reduce the time required for inference.

1. Instead of calculating the output of the nth tree, we calculate the outputs of trees 1-n in series.
2. Instead of passing trees between the clients and servers every round, we train the trees and send them to the clients once at the start of the simulation.

It is difficult to determine whether our submission achieves the best possible performance of a federated version of our Phase 1 model, mainly because the features we were asked to treat as "private" either were not used by our Phase 1 model or had very little impact. However, not knowing which features might actually end up being used by airlines, we chose to federate our model in a way which would generalize well to features with greater importance.

5. What do you see as the most significant remaining work needed to deploy your Phase 2 solution in the real world? For example, coordinating training and/or evaluation, incorporating other private variables, addressing privacy concerns, etc.

We anticipate that the incorporation of different private variables will be the most significant remaining work. The specifics of the ideal federated solution will depend on the nature of these private variables, as well as how features are extracted from the withheld data.

6. Do you have any useful charts, graphs, or visualizations from the process?
N/A.

7. Copy and paste the 3 most impactful parts of code you developed during Phase 2. Explain what each does and how it helped your model.

Constructs a CatBoost model. This is called once for every airline client; each client constructs a model using the data they have available.

```
def construct_tree(
    dataset: Dataset, label: NDArray, n_estimators: int) ->
    CatBoostRegressor:
    """Construct a catboost tree from tabular dataset."""

    train_pool = Pool(np.array(dataset, copy=True), np.array(label,
    copy=True))

    hyper_params = {
    'iterations': n_estimators,
    'learning_rate': 0.15,
    'l2_leaf_reg': 15,
    'loss_function': 'MAE',
    'thread_count': -1,
    'metric_period': 50,
    'max_depth': 9,
    "max_bin": 63,
    'eval_metric': 'MAE',
    }

    tree = CatBoostRegressor(**hyper_params)
    tree.fit(train_pool)

    return tree
```

Extracts a prediction at a single tree index using the CatBoost staged_predict function. The result is fed into a CNN.

```
def single_tree_prediction(
    tree: CatBoostRegressor, n_tree: int, dataset: NDArray
```

```
    ) -> Optional[NDArray]:
    """Extract the prediction result of a single tree in the catboost tree
    ensemble."""
    num_t = tree.tree_count_
    if n_tree > num_t - 1:
        print(
        "The tree index to be extracted is larger than the total number of
        trees."
        )
        return None

    return next(tree.staged_predict(dataset.detach().cpu().numpy(),
    ntree_start=n_tree, ntree_end=n_tree+1))
```

Conducts centralized inference. Used to generate the final submission.

```
all_preds = np.array([])
net, trees = model[airport]

for tree in trees:
    tree_preds = []
    for pred in tree_predictions(tree, features.to_numpy()):
        tree_preds.append([pred])

    tree_preds = np.transpose(tree_preds)
    if len(all_preds) == 0:
        all_preds = tree_preds
    else:
        all_preds = np.append(all_preds, tree_preds, axis=2)

all_preds = torch.from_numpy(all_preds).double().to(device) # convert to
pytorch tensor
all_preds = net(all_preds).squeeze()
partial_submission_format['minutes_until_pushback'] =
all_preds.cpu().detach().numpy()
```

8. Please provide the machine specs and time you used to run your model.

- **CPU (model):** Intel Core i5
- **GPU (model or N/A):** NVIDIA GeForce GTX 780
- **Memory (GB):** 64 GB

- **OS:** Windows
- **Train duration:** 3 hours on provided dataset
- **Inference duration:** Scales depending on size of dataset

9. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?
Our federated model requires much more RAM than our Phase 1 model -- about 50 GB.

10. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?
N/A.

11. How did you evaluate performance of the model other than the provided metric, if at all?
We evaluated performance of our model using Mean Absolute Error, but we chose to train the CNNs using Mean Squared Error.

12. What are some other things you tried that didn't necessarily make it into the final workflow? (quick overview)
We tried experimenting with various training durations (increasing the number of rounds of CNN updates simulated). We also attempted to train simpler CatBoost trees than we had used in our Phase 1 model, but found that this did not improve performance.

13. Are there any other learnings or conclusions from Phase 2 that are not covered in the previous questions that you would like to include? (optional)
N/A.