CIS 4230/5230 Write Up

DAG
Grace Benner, Denise Faria, Anuoluwa Akibu

Model Performance Metrics

Team Model RMSE (validation): 16982.90

Team Model RMSE (train): 15987.90

Number of private updates: 8

Number of global updates: 0

Reduction in global RMSE: 0

Methods

5-7 expected

1. Single Attribute Group Based on High RMSE - Larger Group

Description: Using this method, we picked a group g based only on one attribute (i.e. RAC1P, SEX) and found which groups tended to have higher RMSE in our training predictions. This method was done using exploration of the RMSE, rather than using heuristics we already have about groups. After finding a group with a larger RMSE than the rest, we then trained a new regressor h based only on the subgroup (used train, test splits to avoid overfitting). We used no upsampling, and used some hyperparameter tuning techniques. In this method, we mainly targeted very large group sizes of 80,000+, with some groups being over 50% of observations.

Effectiveness: This method was very effective to get some significant drops in the RMSE. It contributed to 75% of our updates, but lost effectiveness once our model hit under 17,000 validation RMSE. It was so effective at first because we targeted large groups, which lead to larger drops.

# private updates using method: 6

# global updates using method: 0

2. Targeted Correction

Description: Using this method, we picked a group g of significant size based on some prediction interval (i.e. [55000, 60000]) with a higher RMSE. This method was also done using exploration of the RMSE. After finding a group with a larger RMSE than the rest, a new regressor was trained based on the subgroup using all of the training data. No upsampling was used.

Effectiveness: This method was somewhat effective, contributing to the other 25% of our updates. This reduced RMSE by around 338.8, but we could not achieve any more updates using this method after our 2 successful attempts.

# private updates using method: 2

# global updates using method: 0

3. Epsilon Above/Below

Description: Using this method, we picked a group g of significant size based on some overestimation/underestimation by at least/most epsilon (attempts were using epsilon values in the range of [2000,10000] inclusive). After finding a group with a larger RMSE than the rest, a new regressor was trained based on the subgroup using train test splits. No upsampling was used.

Effectiveness: This method was not effective, likely due to the success of methods 1 and 2, which were attempted before this.

# private updates using method: 0

# global updates using method: 0

4. Multi-Attribute Group Based on High RMSE and Heuristics - Smaller Group

Description: Since we seemed to have exhausted large groups, we needed to find smaller groups to really fine tune the groups with remaining high RMSE. For this new method we are looking at combinations of attributes that had noticeably smaller group sizes than in method 1, but still large enough to make an impact on the overall model. Group sizes were around 10,000 to 50,000. Again we looked for high RMSE or just guessed groups that might be more poorly predicted. After finding a group with a larger RMSE than the rest, we then trained a new regressor h based only on the subgroup (used train, test splits to avoid overfitting). We used no upsampling, and used some hyperparameter tuning techniques.

Effectiveness: We did not get any updates using this technique, however we started using this method after having reduced the RMSE by a lot. An issue we had was that our training RMSE was significantly lower than our validation RMSE, so it was hard to know if a given g,h improvement was likely on that set.

# private updates using method: 0

# global updates using method: 0

5. Random Subset

Description: Random subsets is exactly as it sounds, where a classifier is used to randomly select a subset of the training data with a given size, and then an h that performs well on the group is found. We made sure to check the group size of the g to make sure that it wasn't too small. We also did check the RMSE of the randomly selected group, and if it was lower than the overall training RMSE we did not train an h to that group. In the training of the h, we made sure to use train, test splits on the group g to avoid overfitting. We used no upsampling, and used some hyperparameter tuning techniques.

Effectiveness: We did not get any updates using this technique, however we started using this method after having reduced the RMSE by a lot. This method just blindly guesses groups, so it is unsurprising that there was not much lift. It may have been more effective had we tried it before our other methods.

# private updates using method: 0

# global updates using method: 0

Free Response Questions

FRQ1

1. $\varepsilon_(f_0) = 5/5 = 1$
2. $\varepsilon_{g0}(f_0) = 3/3 = 1$
3. $\varepsilon_{g0}(h_0) = 1/3 = 0.3333$
4. $\rho(g_0) = 3/5 = 0.6$
5. $\Delta\epsilon_{g0} = 1 - 0.3333 = 0.6667$
6. Yes, (g_0, h_0) will be accepted because h_0 is a large enough improvement on the error of g_0, and g_0 is a large enough group.
7. $\varepsilon_{g0}(f_1) = \varepsilon_{g0}(h_0) = 1/3 = 0.3333$
8. $\varepsilon(f_1) = 3/5 = 0.6 = \varepsilon(f_0) - (\rho(g_0) * \Delta\epsilon_{g0}) = 1 - (0.6 * 2/3) = 1 - 0.4 = 0.6$

FRQ2

1. (g_1, h_1)
   - Accepted: yes
   - Repairs: (g_0, h_0) is repaired
2. (g_2, h_2)
   - Accepted: no
   - Repairs: n/a
3. (g_3, h_3)
   - Accepted: yes
   - Repairs: n/a

FRQ3

1. Max updates: 5 - If we assume that any update that improves the model accuracy by a non-zero amount is accepted, then the maximum number of updates that we could accept from our starting model f0 would be 5 updates. This is the case because in f0 our model predicts none of the x's correctly, and if each update now predicts one more x correctly, at most we would have 5 accepted updates until the error is 0.
2. Formula for max updates: $\varepsilon(f)/\alpha$ - In the general sense, if we have an initial model f with error ε(f) and require a decrease in α on each update, then the max number of updates possible would be $\varepsilon(f)/\alpha$.

FRQ4

- $g_1 = [0, 0, 1, 1, 1], h_1 = [0, 0, 0, 1, 1]$
- $g_2 = [1, 1, 1, 0, 0], h_2 = [1, 1, 1, 0, 0]$

Assuming we start with y = [1, 1, 1, 1, 1] and f_0 = [0, 0, 0, 0, 0], then if we update with (g_1, h_1) first, the accuracy is 0.4, and if we do second update (g_2, h_2), the accuracy is 1.0. However, if we start with (g_2, h_2), then the accuracy is 0.6, and if we do a second update with (g_1, h_1), then the accuracy will only be 0.8. If we had the repair framework, we would add a repair pointer for g_1, so no matter the order of pairs submitted, it would always run values of g_1 with h_1, and the remaining values in g_2 would go through h_2. Thus, perfect accuracy.

FRQ5

- $g_1 = [1, 0, 1, 0, 0], h_1 = [1, 1, 0, 0, 0]$
- $g_2 = [0, 1, 1, 1, 1], h_2 = [0, 1, 1, 1, 1]$

Assuming we start with y = [1, 1, 1, 1, 1] and f_0 = [0, 0, 0, 0, 0], the first update with (g_1, h_1) results in $f_1 = [1, 0, 0, 0, 0]$ where accuracy is 0.1, and the second update with (g_2, h_2) results in $f_2 = [1, 1, 1, 1, 1]$ where accuracy is 1.0. However, if we update f_0 with (g_2, h_2) first, $f_1 = [0, 1, 1, 1, 1]$ where accuracy is 0.8. When we try to update with (g_1, h_1) next, the PDL model remains unchanged as $\varepsilon_{g_1}(f_1) = \varepsilon_{g_1}(h_1) = \frac{1}{2}$. Thus, only one update occurs with non-perfect accuracy. This implies even when adding a repair framework as done in FRQ4, order matters when introducing updates.

Accepted Updates

Update 1

Method Name: Single Attribute Group Based on High RMSE

Type of update: Private

Code

```
def g(X):
    return X['RAC1P'] == 6
clf = sk.tree.DecisionTreeRegressor(max_depth = 12, random_state = 42)
indices = g(x_train_subset)
clf.fit(x_train_subset[indices], y_train_subset[indices])
h = clf.predict
```

Update 2

Method Name: Single Attribute Group Based on High RMSE

Type of update: Private

Code

```
def g(X):
    return X['RAC1P'] == 1
indices = g(x_train_subset)
reg = sk.ensemble.GradientBoostingRegressor(n_estimators = 500, max_depth = 8,
min_samples_split = 5, learning_rate = 0.01)
reg.fit(x_train_subset[indices], y_train_subset[indices])
h = reg.predict
```

Update 3

Method Name: Single Attribute Group Based on High RMSE

Type of update: Private

Code

```
def g(X):
    return X['SEX'] == 1
indices = g(x_train_subset)
reg = sk.ensemble.GradientBoostingRegressor(n_estimators = 500, max_depth = 12,
min_samples_split = 5, learning_rate = 0.01, loss = "squared_error")
reg.fit(x_train_subset[indices], y_train_subset[indices])
h = reg.predict
```

Update 4

Method Name: Single Attribute Group Based on High RMSE

Type of update: Private

Code

```
def g(X):
    return X['AGEP'] > 50
indices = g(x_train_subset)
reg = sk.ensemble.GradientBoostingRegressor(n_estimators = 500, max_depth = 8,
min_samples_split = 5, learning_rate = 0.01, loss = "squared_error")
reg.fit(x_train_subset[indices], y_train_subset[indices])
h = reg.predict
```

Update 5

Method Name: Single Attribute Group Based on High RMSE

Type of update: Private

Code

```
def g(X):
    return X['MAR'] == 1
indices = g(x_train_subset)
reg = sk.ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 12,
min_samples_split = 5, learning_rate = 0.01, loss = "squared_error")
reg.fit(x_train_subset[indices], y_train_subset[indices])
h = reg.predict
```

Update 6

Method Name: Targeted Correction

Type of update: Private

Code

```
### TARGETED CORRECTION ###

# DEFINING G
value = 45000
epsilon = 6000
binary_labels = abs(y_predictions - value) < epsilon
clf = sk.tree.DecisionTreeClassifier(max_depth = 12, random_state = 42)
clf.fit(x_train, binary_labels)
g = clf.predict

# DEFINING H
clf = sk.tree.DecisionTreeRegressor(max_depth = 12, random_state = 42)
indices = g(x_train)
clf.fit(x_train[indices], y_train[indices])
h = clf.predict
```

Update 7

Method Name: Targeted Correction

Type of update: Private

Code

```
### TARGETED CORRECTION ###

# DEFINING G
value = 60000
epsilon = 10000
binary_labels = abs(y_predictions - value) < epsilon
clf = sk.tree.DecisionTreeClassifier(max_depth = 10, random_state = 42)
```

```
clf.fit(x_train, binary_labels)
g = clf.predict

# DEFINING H
indices = g(x_train)
clf = sk.ensemble.GradientBoostingRegressor(max_depth = 8, n_estimators = 200,
random_state = 42, loss = 'squared_error')
clf.fit(x_train[indices], y_train[indices])
h = clf.predict
```

Update 8

Method Name: Single Attribute Group Based on High RMSE

Type of update: Private

Code

```
def g(X):
    return X['MIL'] > 1
indices = g(x_train_subset)
reg = sk.ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 7,
min_samples_split = 5, learning_rate = 0.1, loss = "squared_error")
reg.fit(x_train_subset[indices], y_train_subset[indices])
h = reg.predict
```