Course Project Description

Background

There are two workers who can perform the same task. But depending on the features of specific tasks, they will complete the task with different qualities. Therefore, we want to distribute the task to the best worker for it.

Data

1. train.csv: The training data includes the completed tasks. Each row contains the features of the task, the worker it was randomly assigned to, and the quality of the result. Higher quality is better.
2. prediction.csv: The prediction data includes the tasks to be distributed. Each row only contains the features of the task.

Problem Statement

1. Create a model to distribute tasks in the prediction data to the best worker and predict the quality of the result.
2. (Bonus) Can you provide interval estimates for your prediction?

Please submit a write up in PDF which includes your analysis and your solutions to the above questions, and a CSV file which includes your results for the test data. The CSV file should have two additional columns, worker and quality, compared to test data, representing the worker you want to assign the task to and the predicted quality.

Course Project Report

Zejian Liu, Rice University

zejianliu15@gmail.com

Abstract

We formulate the problem as a randomized experiment from a causal inference perspective, where quality predictions boil down to the regression analysis of the conditional average treatment effect. After data preprocessing and exploratory data analysis, we tune and evaluate four regression techniques for estimating outcome models of two workers: ϵ-support regression, XGBoost, random forest regression, and random forest quantile regression. Random forest quantile regression is selected among the competing methods, which is able to distribute tasks, predict qualities with good accuracy (Problem 1), and provide interval estimates to address the confidence of prediction (Problem 2).

1 Problem formulation

We first formulate this problem from a causal inference perspective. Let a 20-dimensional vector $X \in \mathcal{X}$ denote features of the task (i.e., the covariate), $T \in \{0, 1\}$ denote the worker that the task is assigned to (i.e., the treatment), and $Y(0), Y(1) \in [-1, 1]$ denote the quality performed by worker_0 and worker_1 (i.e., the potential outcomes). Given a new task with feature $X = x$, we wish to estimate the expected difference between the qualities performed by worker_1 and worker_0:

$$\tau(x) := \mathbb{E}[Y(1) - Y(0)|X = x],$$

which is the conditional average treatment effect (CATE) in causal literature. If $\tau(x) > 0$, then we assign the new task to worker_1, otherwise we assign the task to worker_0.

The training set contains observation (T_i, X_i, Y_i) for task $1 \leq i \leq n$, where $Y_i = T_i Y_i(1) + (1 - T_i)Y_i(0)$. However, we can only observe one of $Y_i(0)$ and $Y_i(1)$. In other words, if the task was assigned to worker_0, we will never now what the quality would be if the same task were assigned to worker_1. Thus, half of the data is missing in this problem. Fortunately, all tasks were randomly assigned to a worker, i.e., the features do not have any effect on the assignment of the task. Figure 1 illustrates the idea of this problem from a causal perspective.

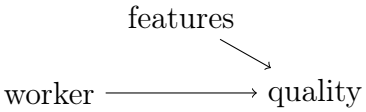


Figure 1: Causal diagrams for worker assignment (T), features (X) and quality (Y).

Thanks to the good properties of a randomized experiment, we are able to calculate $\tau(x)$ via the associational difference:

$$\tau(x) = \mathbb{E}[Y|T = 1, X = x] - \mathbb{E}[Y|T = 0, X = x].$$

We will fit two conditional outcome models, $\hat{\mu}_0(x)$ and $\hat{\mu}_1(x)$, as estimates of $\mathbb{E}[Y|T = 1, X = x]$ and $\mathbb{E}[Y|T = 0, X = x]$, respectively. $\tau(x)$ is then estimated by $\hat{\tau}(x) = \hat{\mu}_1(x) - \hat{\mu}_0(x)$. We formalize the solution to task assignment and quality prediction in Algorithm 1.

Algorithm 1 Algorithm for task assignment and quality prediction

Input Training data (T_i, X_i, Y_i)
Output Task assignment and predicted quality $\hat{Y}_i$ for task i in testing set
Fitting regression models $\hat{\mu}_0(x)$ and $\hat{\mu}_1(x)$ using training data
for task i in testing set **do**
    Compute $\hat{\mu}_0(X_i)$ and $\hat{\mu}_1(X_i)$
    if $\hat{\mu}_0(X_i) > \hat{\mu}_1(X_i)$ **then**
        Assign task i to worker_0
        $\hat{Y}_i = \hat{\mu}_0(X_i)$
    else
        Assign task i to worker_1
        $\hat{Y}_i = \hat{\mu}_1(X_i)$

2 Exploratory data analysis

The training set has 232,483 observations, including 20 features (`x_0`, ..., `x_19`), a column of random task assignment, and an outcome (`quality`). The features consist of 19 continuous variables and 1 categorical variable, and we convert the binary categorical variable to a dummy variable. The outcome is continuous with range $[-1, 1]$. 81,127 tasks are randomly assigned to worker_0 while 151,356 tasks are randomly assigned to worker_1. Figure 2 shows the summary table of features in the training set. We display the histograms of qualities of tasks assigned to worker_0 and worker_1, respectively in Figure 3.

	feature_0	feature_1	feature_2	feature_3	feature_4	feature_6	feature_7	feature_8	feature_9	feature_10
count	232483.000000	232483.000000	232483.000000	232483.000000	232483.000000	2.324830e+05	232483.000000	232483.000000	232483.000000	2.324830e+05
mean	7.310784	22.706030	2280.923329	0.017021	329.840031	6.601858e+05	2.697156	0.001495	0.000967	7.589018e+04
std	3.983856	9.733159	3408.242562	0.999857	735.758682	1.272460e+06	3.218846	0.000972	0.000831	2.728124e+05
min	0.000000	3.000000	100.500000	-1.000000	100.000000	1.005000e+04	0.000000	0.000000	0.000000	0.000000e+00
25%	4.000000	20.000000	980.500000	-1.000000	100.000000	1.334000e+05	1.254390	0.000821	0.000488	4.100000e+03
50%	7.000000	30.000000	1641.500000	1.000000	100.000000	2.507000e+05	1.839238	0.001309	0.000725	1.360000e+04
75%	11.000000	30.000000	2460.000000	1.000000	300.000000	6.072000e+05	2.722157	0.001918	0.001151	5.190000e+04
max	13.000000	31.000000	71765.000000	1.000000	50600.000000	3.296675e+07	117.854280	0.035978	0.020108	2.840220e+07

	feature_11	feature_12	feature_13	feature_14	feature_15	feature_16	feature_17	feature_18	feature_19	quality
count	232483.000000	232483.000000	232483.000000	232483.000000	232483.000000	232483.000000	2.324830e+05	232483.000000	2.324830e+05	232483.000000
mean	0.029862	0.000465	0.000275	2.555226	0.001390	0.003715	6.927992e+04	0.028719	4.533310e-04	-0.060719
std	0.056070	0.000348	0.003025	3.173949	0.000889	0.002698	2.211334e+05	0.048710	3.308627e-04	0.187602
min	0.000006	0.000000	-0.074276	0.010000	0.000002	0.000002	0.000000e+00	0.000014	3.004814e-07	-1.000000
25%	0.005445	0.000235	-0.001126	0.988556	0.000764	0.002083	3.763248e+03	0.006044	2.234893e-04	-0.119875
50%	0.013514	0.000390	0.000229	1.854901	0.001209	0.003063	1.235694e+04	0.015475	3.742163e-04	-0.027167
75%	0.030303	0.000606	0.001612	2.773176	0.001791	0.004508	4.888962e+04	0.031226	5.913193e-04	0.026226
max	1.000000	0.017895	0.076745	124.188380	0.011639	0.073792	1.049743e+07	1.000000	5.942064e-03	1.000000

Figure 2: Summary statistics of training set.

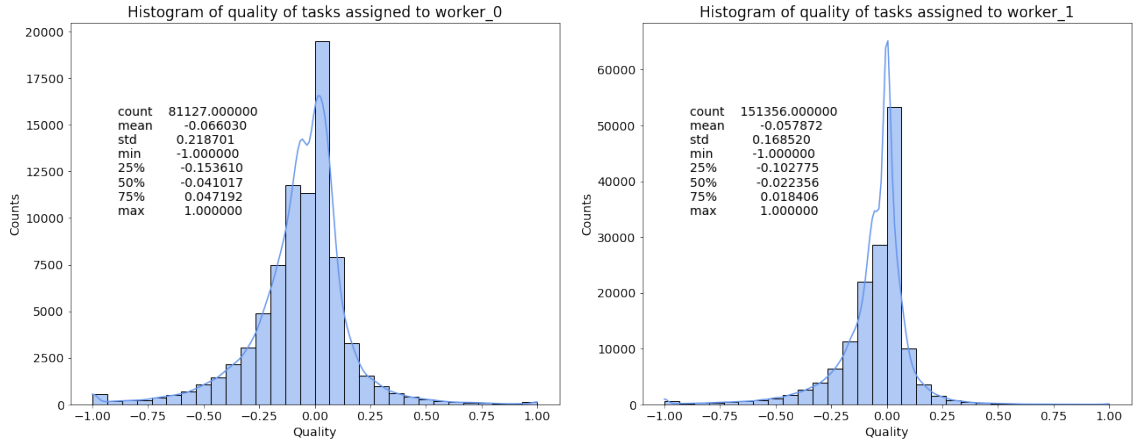


Figure 3: Histograms of quality of tasks assigned to two workers.

To visualize the pairwise relationship between each feature and the outcome, we show the following scatterplots in Figure 4. There seems to be no obvious linear nor nonlinear relationship between the feature and outcome; an exception might be `x_13`, where a linear trend has been observed.

As a last step of the exploratory data analysis, we study the correlation among the features. The correlation heatmap is shown in Figure 5. We do not observe severe multicollinearity among the features.

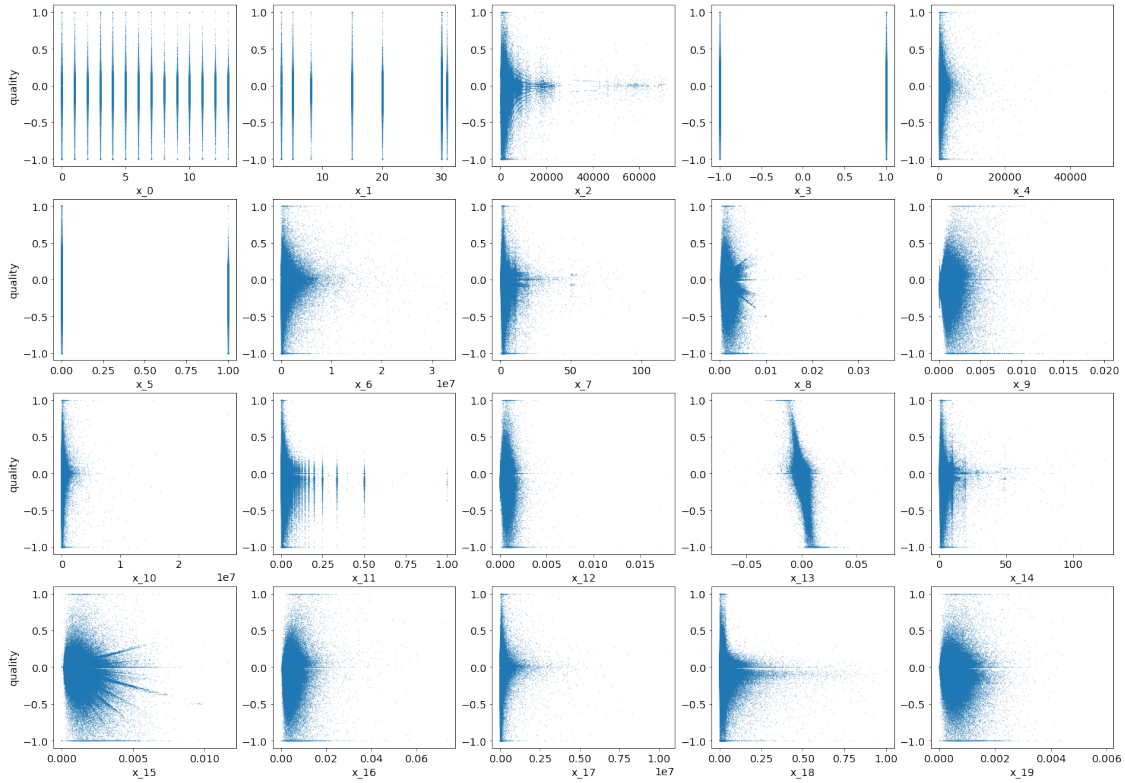


Figure 4: Scatterplots of features and quality.

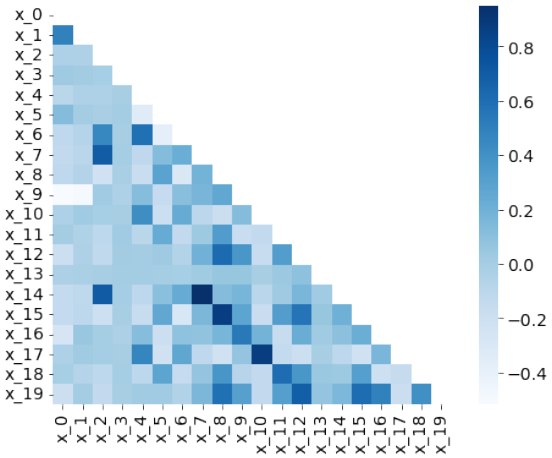


Figure 5: Heatmap of correlation of features.

3 Estimation of outcome models

As mentioned, the problem now bowls down to fitting two conditional outcome models $\hat{\mu}_0(x)$ and $\hat{\mu}_1(x)$ for estimating $\mathbb{E}[Y|T=1, X=x]$ and $\mathbb{E}[Y|T=0, X=x]$. This can be

achieved by training regression models separately on the data that only contain tasks assigned to worker_0 or worker_1. Considering the size of the data, some classical nonparametric regression methods might not perform well (e.g., kernel regression, local polynomial, smoothing splines, etc.). Thus, we utilize machine learning prediction models, such as support vector machine and tree-based techniques.

Specifically, we consider four candidate methods: ϵ-support vector regression (SVR), XGBoost, random forest regression and random forest quantile regression. The candidate methods will be carefully tuned, trained and evaluated on the training set using train-test split.

We divide the training set into two data sets, each contains all tasks completed by worker_0 or worker_1, and are further split into 70% for training and 30% for testing. We compare the performances of the candidate methods according to mean squared error (MSE) and mean absolute error (MAE) of the estimated quality, which are two popular metrics for evaluating regression models. The best method will be applied to predicting the quality and assigning tasks on the testing set.

3.1 ϵ-support vector regression

ϵ-support vector regression is a flexible model that finds an appropriate hyperplane to fit the data.

We first normalize the features and quality, which is a necessary step of data preprocessing for SVR. We use the `svm.SVR` function in the `sklearn` package, and choose the common radial basis function (rbf) as the kernel. A grid search of hyperparameters with 5-fold cross validation is conducted to determine the ϵ in the ϵ-SVR model and the regularization parameter C. The hyperparameters selected are shown in Table 1.

Table 1: Hyperparameters for SVR.

	$\hat{\mu}_0(x)$	$\hat{\mu}_1(x)$
ϵ	0.2	0.2
C	2	2

3.2 XGBoost

XGBoost is a scalable tree-based ensemble technique based on gradient boosting. It has a nice property that allows specification of the model of learning objective. As we transform the output `quality` into $[0, 1]$, the regression can be easily embedded into logistic regression framework for binary classification, where the predicted outcome is given by the predicted probability. We use the `XGBRegressor` function in the `xgboost` package. We select the best the number of trees, the maximum depth of the tree and learning rate via cross validation. The hyperparameters selected are shown in Table 2.

Table 2: Hyperparameters for XGBoost.

	$\hat{\mu}_0(x)$	$\hat{\mu}_1(x)$
learning rate	0.05	0.05
n_estimators	300	100
max_depth	100	100

3.3 Random forest regression

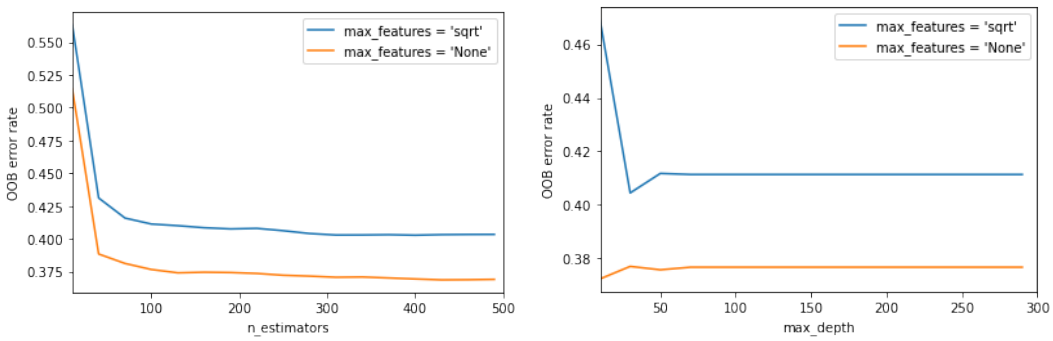
Random forest regression is an ensemble method that fits a number of decision trees on various sub-samples of the data and uses averaging to improve the predictive accuracy and reduce over-fitting.

Scaling of the features is not necessary for random forests, and it is flexible to handle both continuous and categorical inputs. We use the `ensemble.RandomForestRegressor` function in the `sklearn` package. As for hyperparameter tuning, we determine whether to use warm start and bootstrap samples via 5-fold cross validation.

In addition, out-of-bag (OOB) error is an efficient criteria for hyperparameter tuning. When trained using bootstrap aggregation, one-third of the samples can be used to calculate the validation error (i.e., the out-of-bag error), which requires less computation and allows one to test the performance as the method being trained. We select the number of features for split, the number of trees and the maximum depth of trees according to OOB errors. Figure 6 displays the OOB errors under different settings. We can see that max_features = 'None' produces lower OOB errors than max_features = 'sqrt'. For both models the errors stabilize when n_estimators = 300 and max_depth = 100. The hyperparameters selected are summarized in Table 3.

Table 3: Hyperparameters for random forest regression.

	$\hat{\mu}_0(x)$	$\hat{\mu}_1(x)$
bootstrap	True	True
warm_start	False	True
max_features	auto	auto
n_estimators	300	300
max_depth	100	100



(a) n_estimators for $\hat{\mu}_0(x)$ (b) max_depth for $\hat{\mu}_0(x)$

(c) n_estimators for $\hat{\mu}_1(x)$ (d) max_depth for $\hat{\mu}_1(x)$

Figure 6: Hyperparameter tuning using OOB errors.

3.4 Random forest quantile regression

The last method we consider is random forest quantile regression, which adopts a different optimization scheme compared with random forest regression. Instead of focusing on the conditional mean, quantile regression aims to estimate the conditional quantile of the outcome variable. Specifically, random forest quantile regression fits $\tilde{\mu}_{t,q}(x)$ that estimates the quantile y at percentile q where

$$F(Y = y|T = t, X = x) = q$$

for $t = 0, 1$ and $q \in (0, 1)$.

There are some advantages of quantile regression over mean regression. Quantile regression is a robust to both observational errors and outliers in the data. In addition, while median regression ($\tilde{\mu}_{t,0.5}$ with $q = 0.5$) can be used as a point estimate as opposed to mean regression, the predictions based on other quantiles ($(\tilde{\mu}_{t,\alpha}, \tilde{\mu}_{t,1-\alpha})$ for some level α) automatically provide interval estimates for uncertainty quantification.

We use the `RandomForestQuantileRegressor` function in the `quantile_forest` package. The same choices of hyperparameters are used as in random forest regression.

3.5 Method evaluation

The training set is split into 70% for training and 30% for testing. We train and evaluate the prediction performances of the four methods. The MAE and MSE are reported in Table 4.

Table 4: Performance of quality prediction of candidate methods.

	worker_0		worker_1	
	MAE	MSE	MAE	MSE
SVR	0.074	0.114	0.073	0.116
Random forest	0.072	0.109	0.074	0.113
Random forest quantile	0.071	0.111	0.070	0.116
XGBoost	0.075	0.113	0.075	0.116

Random forest quantile regression has the best performance in terms of MAE while random forest regression has the lowest MSE. We choose random forest quantile regression as our final outcome model due to its relative good performance and the ability to provide interval estimates (as the answer to Problem 2).

4 Prediction and interval estimation

We train random forest median regression ($q = 0.5$) to obtain point estimates for quality, of which lower and upper interval estimates are provided by quantile regression with $q = 0.025$ and $q = 0.975$. The algorithm is stated as below.

Algorithm 2 Algorithm for task assignment, quality prediction and interval estimation

Input Training data (T_i, X_i, Y_i)
Output Task assignment, predicted quality $\hat{Y}_i$, and interval estimates $(\hat{Y}_{i,L}, \hat{Y}_{i,U})$
Fitting random forest quantile regression models $\tilde{\mu}_{0,q}(x)$ and $\tilde{\mu}_{1,q}(x)$ for $q = 0.025, 0.5, 0.975$
for task i in testing set **do**
  Compute $\hat{\mu}_0(X_i)$ and $\hat{\mu}_1(X_i)$
  if $\hat{\mu}_{0,0.5}(X_i) > \hat{\mu}_{1,0.5}(X_i)$ **then**
    Problem 1: Assign task i to worker_0
    Problem 1: Predicted quality $\hat{Y}_i = \hat{\mu}_{0,0.5}(X_i)$
    Problem 2: Interval estimates $(\hat{Y}_{i,L}, \hat{Y}_{i,U}) = (\hat{\mu}_{0,0.025}(X_i), \hat{\mu}_{0,0.975}(X_i))$
  else
    Problem 1: Assign task i to worker_1
    Problem 1: Predicted quality $\hat{Y}_i = \hat{\mu}_{1,0.5}(X_i)$
    Problem 2: Interval estimates $(\hat{Y}_{i,L}, \hat{Y}_{i,U}) = (\hat{\mu}_{1,0.025}(X_i), \hat{\mu}_{1,0.975}(X_i))$

The feature importances of two models for $\tilde{\mu}_{0,q}(x)$ and $\tilde{\mu}_{1,q}(x)$ are visualized in the following Figure 7. It is shown that `x_13` has the dominating importances in both models, which confirms our finding in exploratory data analysis.

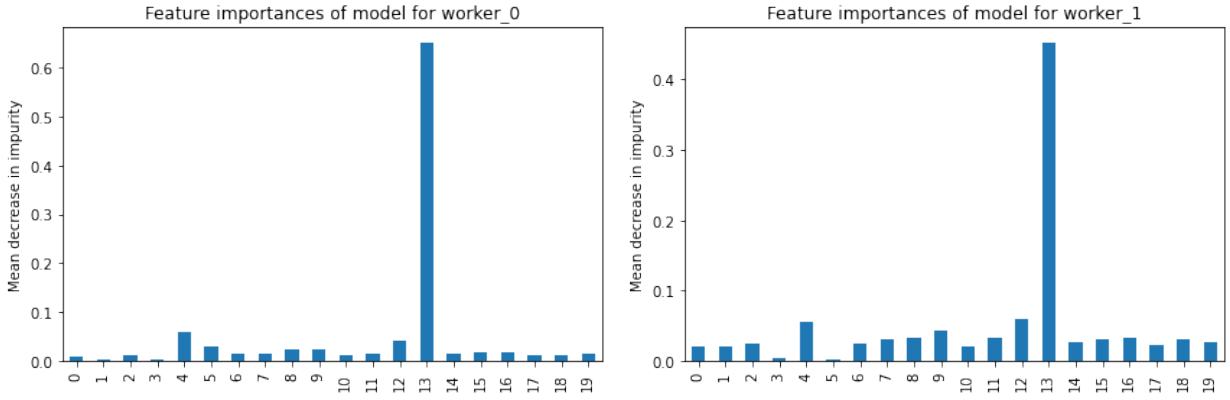


Figure 7: Feature importances of two models.

Finally, we show the summary of our predictions for quality and 95% interval estimates.

	quality	lower	upper
count	85216.000000	85216.000000	85216.000000
mean	-0.031712	-0.242568	0.103701
std	0.145745	0.215830	0.126543
min	-1.000000	-1.000000	-1.000000
25%	-0.080581	-0.324126	0.041832
50%	-0.013160	-0.197019	0.074618
75%	0.022942	-0.113314	0.128972
max	1.000000	1.000000	1.000000

Figure 8: Summary for quality predictions and interval estimates.