Freie Universität Berlin

Aktuelle Forschungsfragen der empirischen Finanz- und Wirtschaftspolitik

Machine Learning

# Tree Based Algorithms to Predict Gross Hourly Wages

*Author*
Roman Schulze

*Supervisor*
Prof. Dr. Viktor Steiner

November 5, 2020

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| cp | complexity parameter |
| e.g. | for example |
| i.e. | in other words |
| GBM | Gradient Boosting Machine |
| MSE | Mean Sqared Error |
| OOB | out-of-bag |
| OLS | Ordinary Least Squares |
| RSS | Residual Sum of Sqaures |
| SOEP | Socioeconomic Panel |
| w.r.t. | with respect to |

# List of Symbols

| | |
|---|---|
| $\alpha$ | complexity parameter |
| $\lambda$ | shrinkage parameter |
| $H_i$ | Annual working hours of individual $i$ |
| $N$ | Number of observations |
| $R_m$ | predicted response of $m$th terminal node |
| $x_i$ | feature, predictor |
| $y_i$ | natural logarithm of gross hourly income |
| $Y_i$ | gross yearly income of individual $i$ |

# 1 Introduction

The Big Data Boom in recent years is one of the main reasons why statistical learning methods based on artificial intelligence such as machine learning are a widely used tool in modern data analysis. The expression machine learning dates back to Samuel (1959) and is usually used for pattern recognition and classifying data correctly. A famous example where machine learning has been applied is automated text categorization (Sebastiani (2002)) .
Machine learning can be divided into two major groups, which are categorized as supervised and unsupervised learning. Given a dataset containing many features and a certain target variable, in supervised learning an algorithm is trained using the features to predict the target variable. In unsupervised learning, however, there is no specific target variable. The goal in unsupervised learning, therefore, is to identify an underlying structure of the dataset (Goodfellow et al. (2016)).
This paper will seek to analyze the accuracy of the predicted gross hourly wages of german employees, based on SOEP data for the survey year 2015. For this investigation four different supervised machine learning algorithms were used, all of them tree based methods. Based on a single regression tree, the ensemble methods bagging, random forest and gradient boosting machine will be applied. All methods are implemented with the statistical programming language *R*. The structure of the paper is as follows: first of all the reader will be introduced to the methodology. Second, the dataset and the model selection are explained. These are then followed by the results and the conclusion.

# 2 Methodology

Starting with explaining how a regression tree works, this section will continue by introducing the concepts of Bagging, Random Forrest and GBM. These concepts are called ensemble learners because, as indicated by the name, many regression

trees are trained to make a prediction, rather than just one. The explanation of the concepts is largely informed by James et al. (2014).

## 2.1 Regression Tree

Decision Trees are a powerful tool for making predictions. In comparison to the classical approach of a linear regression, decision trees are also able to produce good results when the relationship between predictor $y_i$ and predictors $x_i$ is not linear.
Another advantage of decision trees is that they are easy to interpret.
The idea goes back to Breiman et al. (1984). A regression tree is applied to problems, where the explanatory variable is continuous.
The basic idea on how a regression tree works is fairly simple and follows a scheme, which is called recursive binary splitting. Starting with the whole set of features $x_J$ and all observations $N$, the algorithm splits the set of observations into two daughter nodes. The splitting point $t$ is chosen by minimizing a loss function. In case of a regression tree the loss function is given by the MSE. Hence, the objective becomes

$$\min_t MSE = \frac{1}{N_l}\sum_{i\in l}(y_i - \overline{y}_l)^2 + \frac{1}{N_r}\sum_{i\in r}(y_i - \overline{y}_r)^2. \tag{2.1}$$

For the resulting two daughter nodes, the same procedure applies. The process continues until the MSE gain becomes too small. This is the point, where the construction of the tree is completed.
Since the algorithm minimizes the MSE at each node of the tree the most important feature at the initial node (or root node). The prediction is given as the mean within the respective terminal nodes at the bottom of the tree (Breiman et al. (1984)).

### 2.1.1 Pruning

A problem that arises in many applications, is that trees easily overfit. Overfitting is a consequence of fitting the variation in the residuals which leads to a good performance on the training data, but then fails to produce good results on test data. The underlying reason is that trees are usually grown to deep, i.e. the model structure is too complex (James et al. (2014)). To avoid overfitting, it is common practice to prune trees by adjusting the loss function to take the following form

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|. \tag{2.2}$$

The squared term in brackets in 2.2 is simply the MSE within the $m$th leaf node. The pruning component is the latter part, where parameter $\alpha$ is some positive value and $|T|$ is defined as the number of leaf nodes. For $\alpha$ equal to zero the pruned tree and the deep grown tree do not differ. However, if $\alpha$ takes a positive value, the number of leaf nodes has to decrease to yield a better performance than the unpruned tree. Pruning generally reduces the variance, yet it leads to an increase in the bias (James et al. (2014)).

## 2.2 Bagging

In contrast to using a single regression tree, the bagging algorithm by Breiman (1996) constructs an ensemble of $B$ regression trees with each tree being trained only on a subsample of the training data. Each subsample is constructed using bootstrapping methods so that each subsample only includes a fragment of all available training data observations. It might be, that some observations occur twice in a subsample, while others are left out completely.
Taking the number of trees $B$ as given, for each tree the prediction procedure works in the same way as for the regression tree. However, each tree will come up with a different prediction for observation $i$ as each subsample is constructed randomly. Eventually the final prediction $\hat{f}_{bag}(x)$ for an observation is given by averaging over all $B$ subsamples

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x), \tag{2.3}$$

where the prediction for each bagged tree is given by $\hat{f}^{*b}(x)$. In comparison to a single regression the bagged trees are not pruned. This implies that each bagged tree is having a high variance. A high variance means, that splitting a dataset randomly into two subsets the resulting trees for each subset might differ. However, taking the average over all $B$ trees leads to a reduction in the variance (James et al. (2014)).

## 2.3 Random Forest

The random forest algorithm introduced by Breiman (2001) can be considered as an extention of the bagging algorithm. In the first place a forest of 1 to $B$ regression trees is grown. Thereby each of the trees is grown independently using bootstrapping methods. However, compared to bagging each node only a subset ($p$) of all available features ($m$) is taken into account when making a split. If the target variable is continuous a common approach is to choose $p=m/3$. This additional component implies, that features, that do not play an important role in bagging or regression trees become more of a chance. The result is that each of the constructed trees will look quite different than the other ones. Hence, a random forest decorrelates the ensemble of trees compared to bagging. Consequently the decrease in the variance is even higher.

## 2.4 Gradient Boosting Machine

Gradient boosting machine is further ensemble machine learning algorithm. The GBM algorithm by Friedman (2002) also works by constructing an ensemble of trees. However, in the case of GBM each tree is not grown independently of each other, rather than constructed sequentially. This means, that each new tree is using the information of the previously grown tree. Furthermore, instead of fitting the target variable $y_i$, the algorithm works by fitting the residuals. Therefore GBM follows the approach of error-based learning and tends to learn slowly. Consequently GBM usually uses more trees compared to bagging and random forest [1].

# 3 Estimation

## 3.1 Data

The analysis is based on SOEP (Socioeconomic Panel), a longitudinal representative survey which started in 1984. Divided into several datasets, each of them provides different information, e.g. about individual or household charac-

[1] The mathematics behind the algorithm are explained in more detail in James et al. (2014)

teristics. This analysis uses the Person-related Status and Generated Variables dataset (pgen) and the Extended Income Information dataset (pequiv) are used. The study focuses on the survey year 2015. The analysis focuses only on people who are currently employed, i.e. individuals with a positive income. The target variable $y_i$ is the natural logarithm of the gross hourly income and is constructed as

$$y_i = log(\frac{Y_i}{H_i}). \tag{3.1}$$

$Y_i$ is the gross annual income of individual $i$, while $H_i$ measures the number of annual hours worked in the corresponding year. To control for implausible values of income the lower and upper percentile are removed.
12 variables are included as features. Five of them are binaries, four variables are continuous and three are categorical. Table 3.1 lists all variables that are included and provides information about the distribution of the data.

| Variable | N | Mean | St. Dev. | Min | Pctl(25) | Pctl(75) | Max |
|---|---|---|---|---|---|---|---|
| Salary | 10,952 | 2.7 | 0.5 | 1.0 | 2.4 | 3.1 | 4.1 |
| Male | 10,952 | 0.5 | 0.5 | 0 | 0 | 1 | 1 |
| Age | 10,952 | 44.3 | 10.4 | 18 | 37 | 52 | 65 |
| West | 10,952 | 0.8 | 0.4 | 0 | 1 | 1 | 1 |
| Married | 10,952 | 0.7 | 0.5 | 0 | 0 | 1 | 1 |
| Children | 10,952 | 1.0 | 1.1 | 0 | 0 | 2 | 10 |
| Hours | 10,952 | 36.3 | 12.3 | 1 | 30 | 43 | 80 |
| Civil | 10,952 | 0.3 | 0.4 | 0 | 0 | 1 | 1 |
| Education | 10,952 | 12.9 | 2.8 | 7.0 | 11.0 | 15.0 | 18.0 |
| Size | 10,952 | 4.9 | 1.9 | 1 | 4 | 7 | 8 |
| LearnedJob | 10,952 | 0.6 | 0.5 | 0 | 0 | 1 | 1 |
| Occupation | 10,952 | 4.0 | 2.6 | 1 | 2 | 7 | 10 |
| Tenure | 10,952 | 10.8 | 10.2 | 0.1 | 2.5 | 16.8 | 48.8 |

**Table 3.1:** *Descriptive statistics*

The feature Occupation comprises 10 categories, based on "Erikson and Goldthorpe Class Category" [2]. The categorical variable size splits the number of employees per firm into eight categories. Furthermore tenure, the actual working hours (including overtime), the number of children serve as predictors. The total sample comprises 10952 individuals.

[2]The variable is described in table .2

## 3.2 Modelselection

In the first place the original dataset was randomly divided into a training and a test dataset, where the latter includes 30% of the overall observations.
The estimation of the corresponding models is performed in $R$ using the following packages: $rpart$, $randomForest$ and $gbm$ for the respective methods.

### 3.2.1 Parameter Optimization

First the algorithms were trained using the default values. Subsequently, these were then tuned by looking for the best set of hyperparameters. Tuning the models hyperparameters usually leads to an improvement in the predictive performance of a certain algorithm compared to just using the default values (Witten et al. (2011)).
One approach to identify the optimal set of hyperparameters is to perform a grid search. A grid search is an exhaustive searching process, based on a predefined interval that yields the best hyperparameter combination. The best parameter constellation is determined by minimizing

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2. \tag{3.2}$$

Here $y_i$ is the actual value of the target variable and $\hat{f}(x_i)$ is the prediction of the respective algorithm. The crucial part about tuning is that the evaluation is not done on the training data, because we want to know how good the algorithms perform on unseen observations. However, to get a feeling for how good an algorithm with changing parameters performs on unseen data, before applying it to the test data, the trick is to split the training data to obtain an additional validation dataset. At this point it is worth noting that there are three datasets: training, validation and test. Now, for each parameter combination the model is trained using the training data, while the performance is evaluated on the validation dataset. The hyperparameter constellation yielding the best performance on the validation dataset is now used for making a prediction on the unseen test dataset (Witten et al. (2011)).
However, in case of Bagging and the Random Forest it is also possible to make use of the out-of-bag observations, rather than constructing an additional validation dataset for the tuning process. OOB are observations that were randomly kept out during the training process. For each tree around one third of the training

observations are kept out due to bootstrapping and can be used as a validation set. The OOB performance is a good indicator for the performance on the unseen test data (James et al. (2014)). For the regression tree, a 10-fold-cross-validation is performed to tune hyperparameter $\alpha$. k-fold cross-validation is a validation procedure, which divides the training data into k sets. For the $k$th set a prediction will be made based on the the k-1 sets. This is performed k times, so that each of the k sets serves as a validation set once. Finally the average of the k performances is used to choose the optimal value for $\alpha$ (Goodfellow et al. (2016)). Table 3.2 summarizes which hyperparameters are taken into account for the corresponding algorithms and by which method they are tuned.

| **Algorithm** | **Hyperparameter(s)** | **Method** |
|---|---|---|
| Regression Tree | $\alpha$ | 10-fold cross-validation |
| Bagging | $nodesize, ntrees$ | OOB |
| Random Forest | $nodesize, ntrees, mtry$ | OOB |
| GBM | $\lambda, interaction.depth, n.trees$ | OOB |

**Table 3.2:** *Hyperparameters of algorithms that are tuned.*

To optimize the performance of the single regression tree the algorithm searches for the optimal complexity parameter $\alpha$ to prune the tree.
The tuning parameters in case of bagging are given by *nodesize* and *ntree*. $Nodesize$ controls for the minimum amount of terminal nodes. A larger value results in smaller trees and takes less computational time. The parameter *ntree* determines the number of trees that are constructed. Tuning the random forest additionaly includes the parameter $mtry$, which controls for the number of variables tried at each node to make a split.
The tuning of the GBM algorithm is based on three hyperparameters. These are the shrinkage parameter $\lambda$, the interaction depth and the number of trees. The shrinkage parameter regulates the learning rate of GBM. How many splits are performed in each tree is controlled by the interaction depth. Finally, the number of trees is also included of the grid search (James et al. (2014)).

# 4 Results

## 4.1 Regression Tree

Figure 4.1 is giving the optimal value to prune the tree. Notice, that $\alpha$ is set to zero before pruning. A positive value for $\alpha$ leads to a shorter subtree. The bigger the value for $\alpha$ , the shorter will be the resulting subtree. The cross validation approach checks the performance for different values of $\alpha$ and chooses $\alpha$ by minimizing 2.2.
On the bottom of the x-axis the different values for complexity parameter $\alpha$ are visible, ranging from infinity to zero. The upper x-axis indicates how many leaf nodes $|T|$ coincide for different values of $\alpha$. The y-axis gives the error of the 10-fold cross-validation.

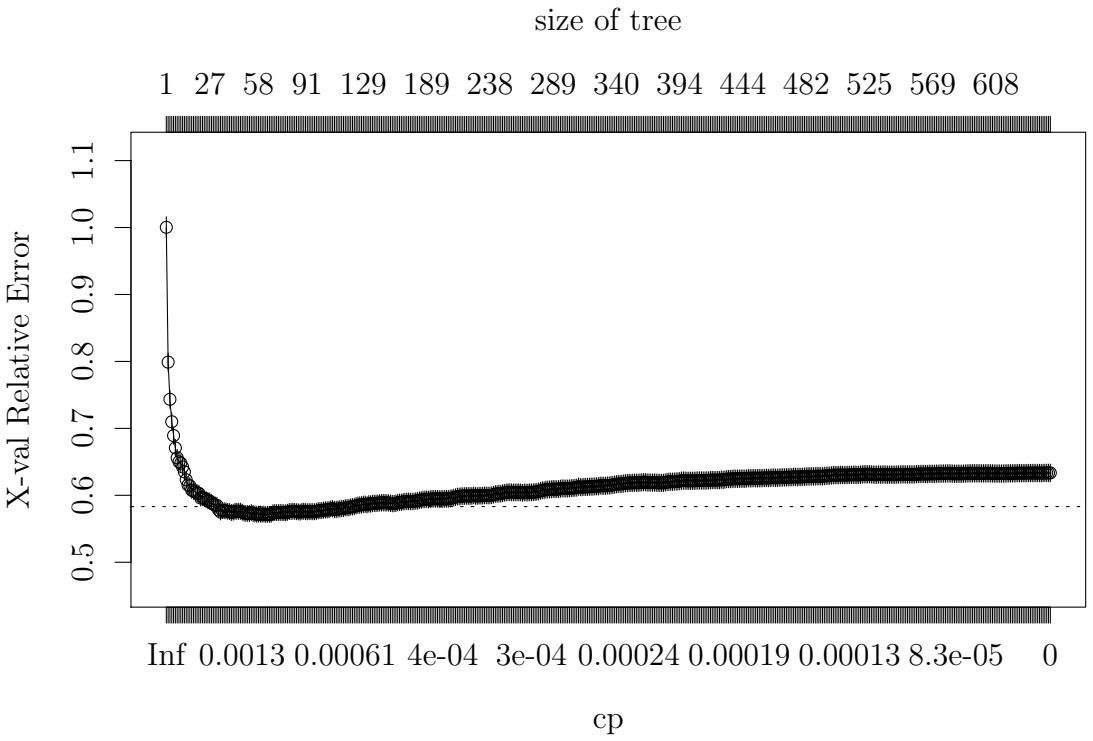


**Figure 4.1:** *Choosing complexity parameter $\alpha$ by minimizing the error of the 10-fold cross-valdidation.*

In the first place the error decreases rapidly, resulting in a minimum at around 60 leaf nodes and then starts to increase again. The optimal value for$\alpha$ is given by 0.001.
Due to the fact that the pruned tree is still quite deep figure 4.2 just shows the

first three layers.

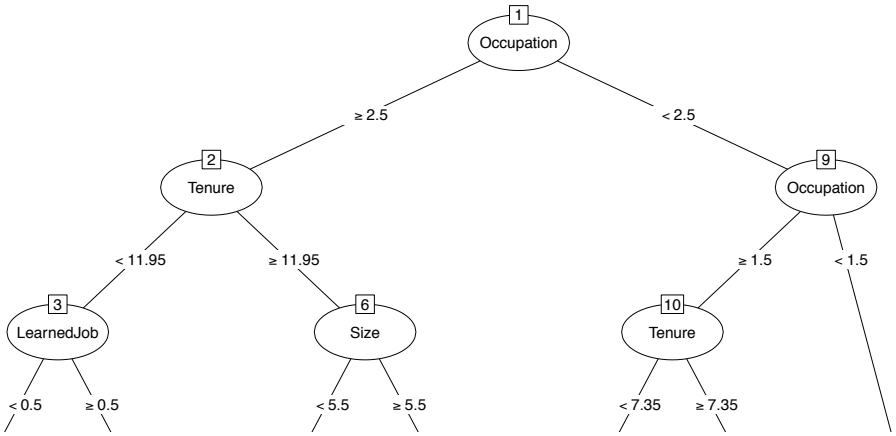


**Figure 4.2:** *Pruned Tree: The graphic shows the first three layers of the pruned regression tree.*

The most important feature is Occupation, where the cut-off point is 2.5, i.e. all observations being in a Occupation category bigger than two are sent to the left. For all further splits the same logic applies.
Figure 4.3 plots the predicted against the actual values and gives an impression how good the regression tree performs.

The predicted mean of the leaf nodes corresponds to the values under the vertically dotted lines. On the orange line predicted and actual values coincide. Ideally all points are on that line or very close, which implies a low error. Hence, it looks like there is room for improvement.

## 4.2 Bagging and Random Forest

This section presents the results of the bagged model and random forest.

### 4.2.1 Bagging

Table 1 presents the *R*-Output of the estimated bagged model. The summary shows that all 12 features are taken into account to make split at each node. For Bagging the optimal set of hyperparameters is given by $nodesize$=18 and $ntree$=500.
The OOB fit is given by the (pseudo) $R^2$, which states that the bagged tree is able to explain 50% of variance in the target variable. Additionally the OOB mean of

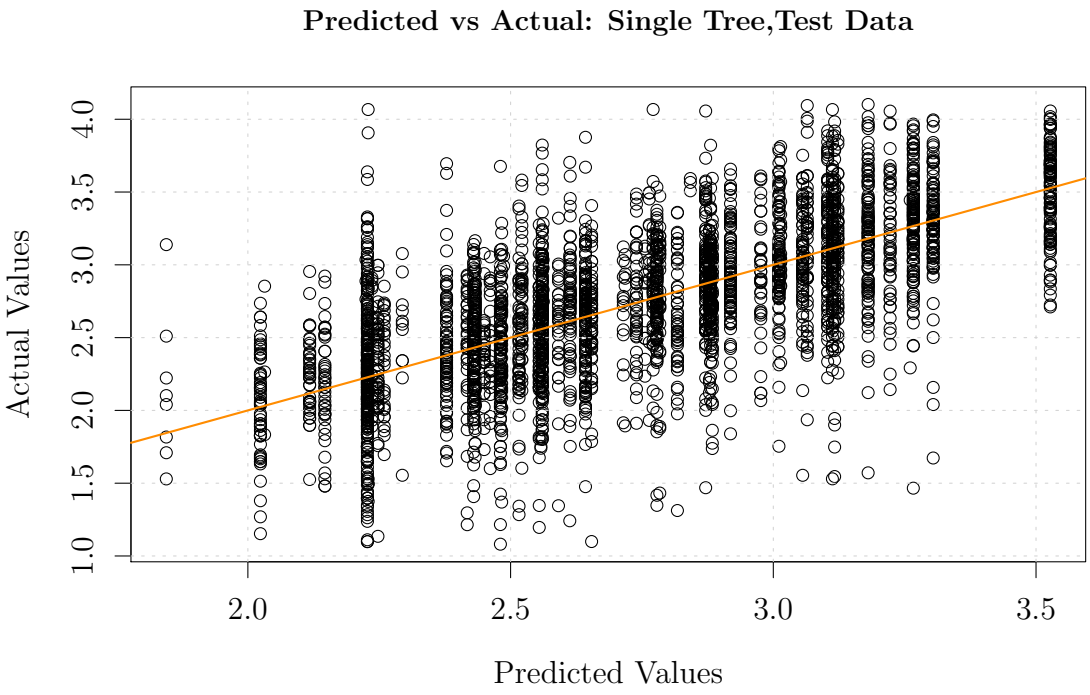


**Figure 4.3:** *The graph plots the actual against the predicted values.*

squared residuals is given by 0.141.

A comprehensive summary on the feature importance is given by figure 4.4.

Features, having a high value are more important in predicting the response (James et al. (2014)). Similar to the regression tree occupation is the most important predictor strongly dominating the other features. Further important features in predicting the hourly wage are tenure, the size of the firm and the number of years of eduacation. However, learnedjob, which appears in the top layers of the regression tree does not play an important role in bagging.

### 4.2.2 Random Forest

Table 2 presents the *R*-Output of the estimated random forest model. The optimal values for the tuned hyperparameters are given by $nodesize$=12 ,$ntree$=500, $mtry$=3. The measures of fit indicate a slight improvement compared to the bagged model. Figure 4.5 shows which features are important in case of the Random Forest.

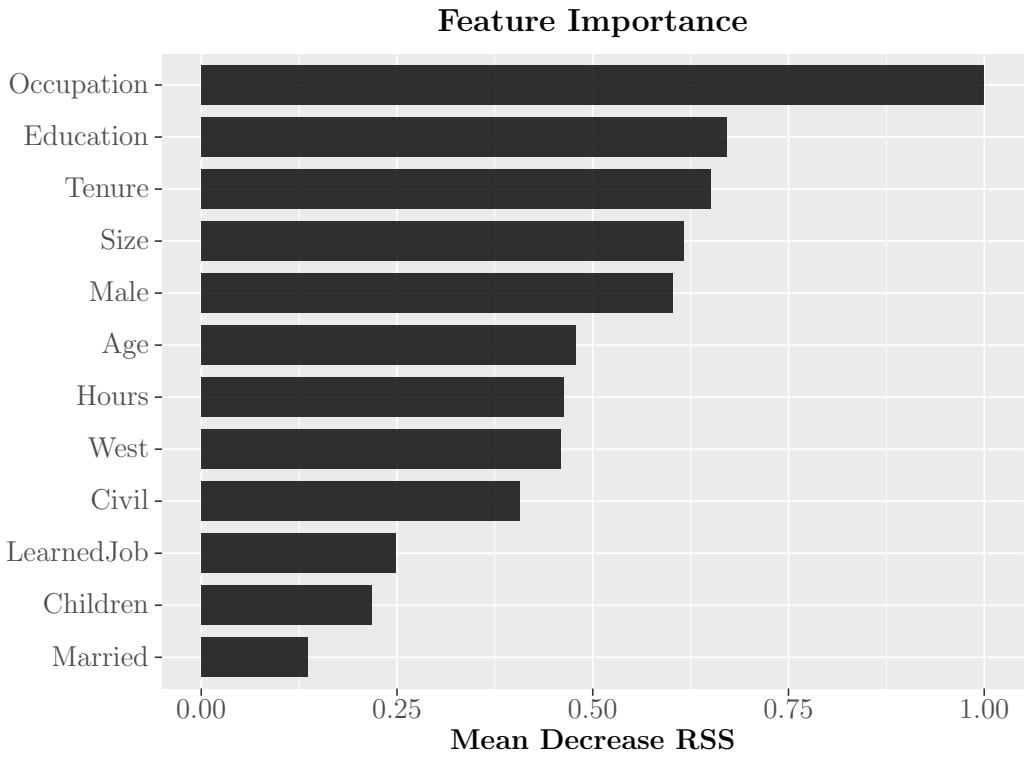


**Figure 4.4:** *Feature Importance of Bagging*

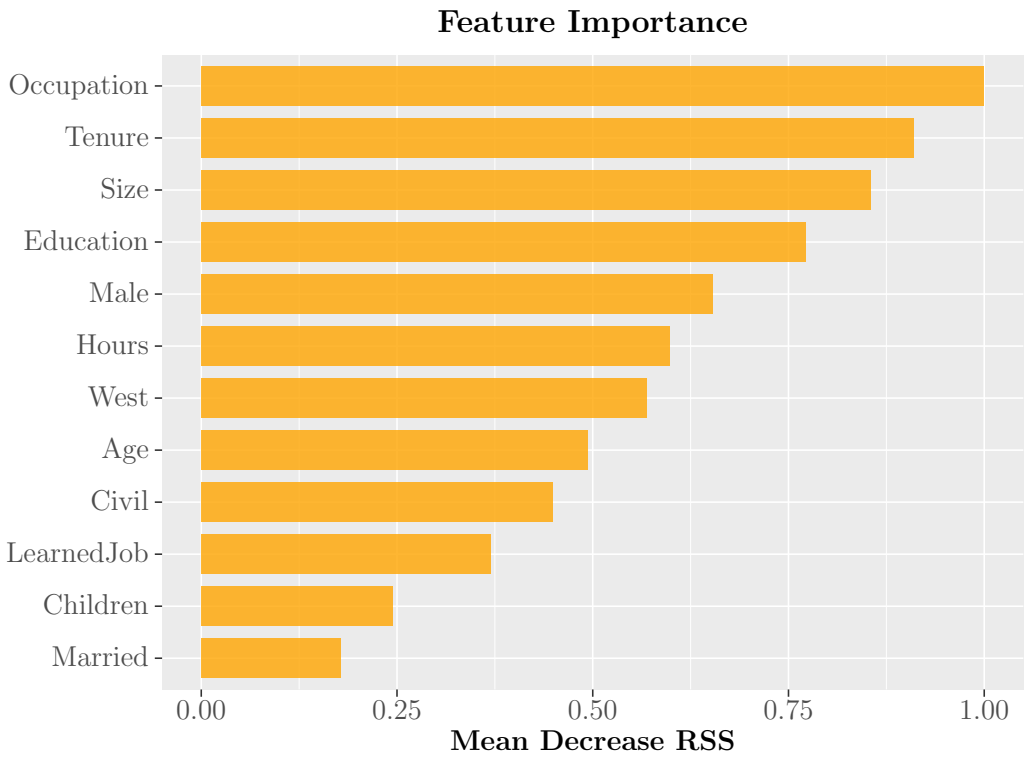


**Figure 4.5:** *Feature Importance of Random Forest*

Compared to the bagged model the relative influence of occupation in reducing the RSS has decreased. The reason is that bagging considers occupation at each

node of the 500 trees as a possible feature to make a split. However, in the case of the random forest on average 3/4 of the splits do not consider occupation or another of the strong predictors (James et al. (2014)). That is why the relative influence plot of the random forest looks more balanced. Furthermore the order of the stronger predictors has changed a bit.

The graphic 4.6 relates the OOB-MSE and the tree size of the corresponding algorithm.

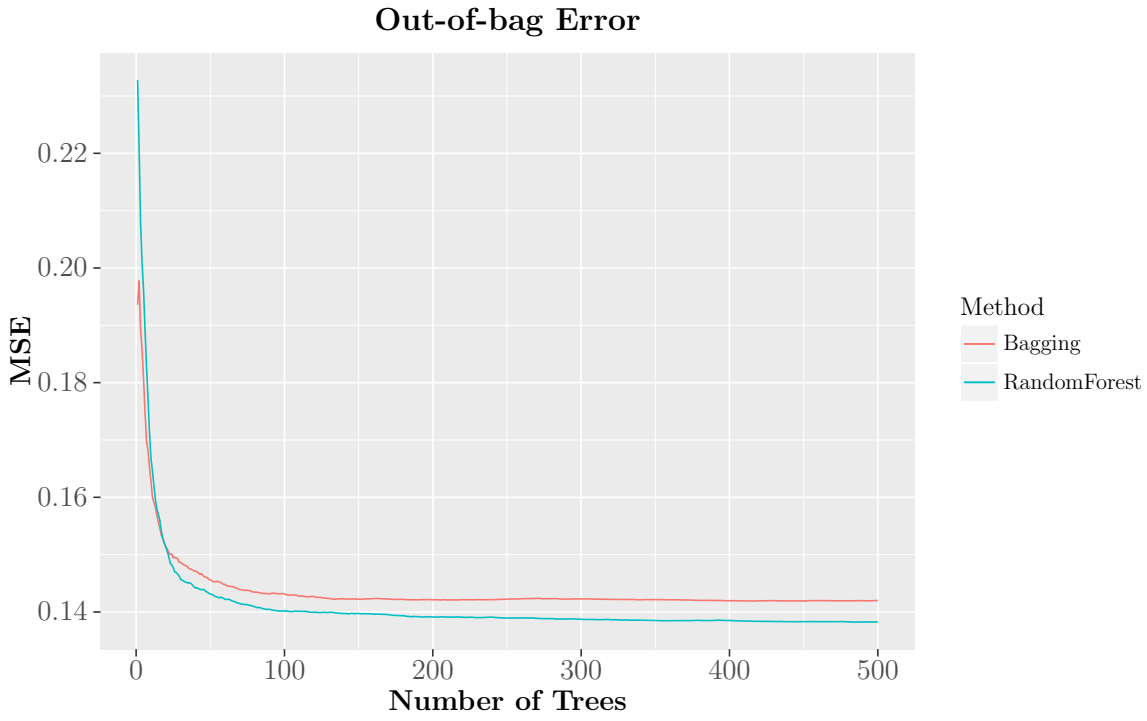


**Figure 4.6:** *OOB: MSE vs Number of Trees for bagging and random forest*

In both cases the MSE and the number of trees show a negative relationship, i.e. an increase in the number of trees leads to a reduction in the OOB-MSE. For few trees in both cases the Error is above 0.2 declining rapidly to a value beneath 0.15. The MSE smoothes out after around 100 trees, resulting in a little lower OOB-MSE for the random forest.
As the OOB results of the random forest are slightly better I expect the Random Forest to yield a better prediction result, when it comes to the prediction on the test dataset.

## 4.3 Gradient boosting Machine

Eventually this section presents the results of the GBM algorithm. Table 3 delivers the output for the trained model. The shrinkage parameter $\lambda$ takes a value of 0.001 and the interaction depth is equal to seven. The number of trees in the GBM model is much higher than in the bagged model and the random forest. Hence, the algorithm learns very slowly.
Similar to bagging and random forest it is possible to obtain a table with the relative influence of each predictor (figure .1 in Appendix). The ranking of the feature importance is similar to the one from random forest. Yet, the relative influence of Occupation strongly dominates the other features.
Furthermore the GBM algorithm allows the construction of partial dependence plot that gives the marginal effect of chosen features on the predicted response variable.

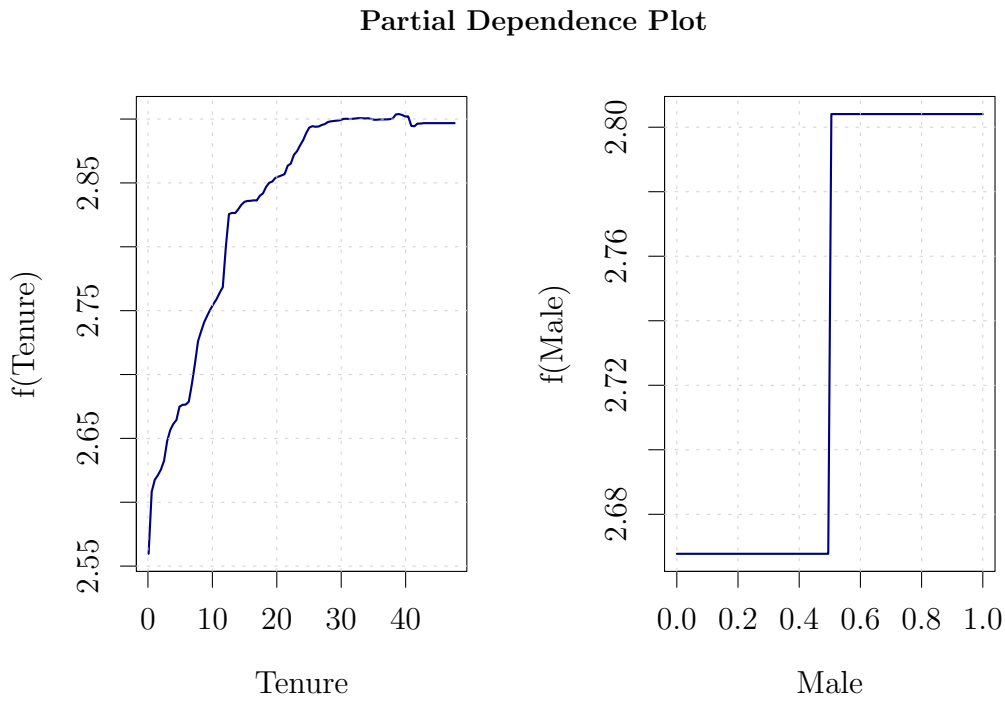


**Figure 4.7:** *Partial Dependence Plot for Tenure and Male:*

This happens by averaging out the effects of all the other predictors (James et al. (2014)). An increase in tenure, the GBM model predicts an increase in gross hourly wage on average. Furthermore the model predicts men to have a higher hourly wage compared to women.

## 4.4 Performance on test data

To evaluate the performance of the ML algorithms each algorithm was used to make a prediction on the test dataset. To make the performances comparable the MSE has been calculated for each of the predictions. A lower value of the MSE implies a better performance. The histogram in figure 4.8 shows the results for the different algorithms.

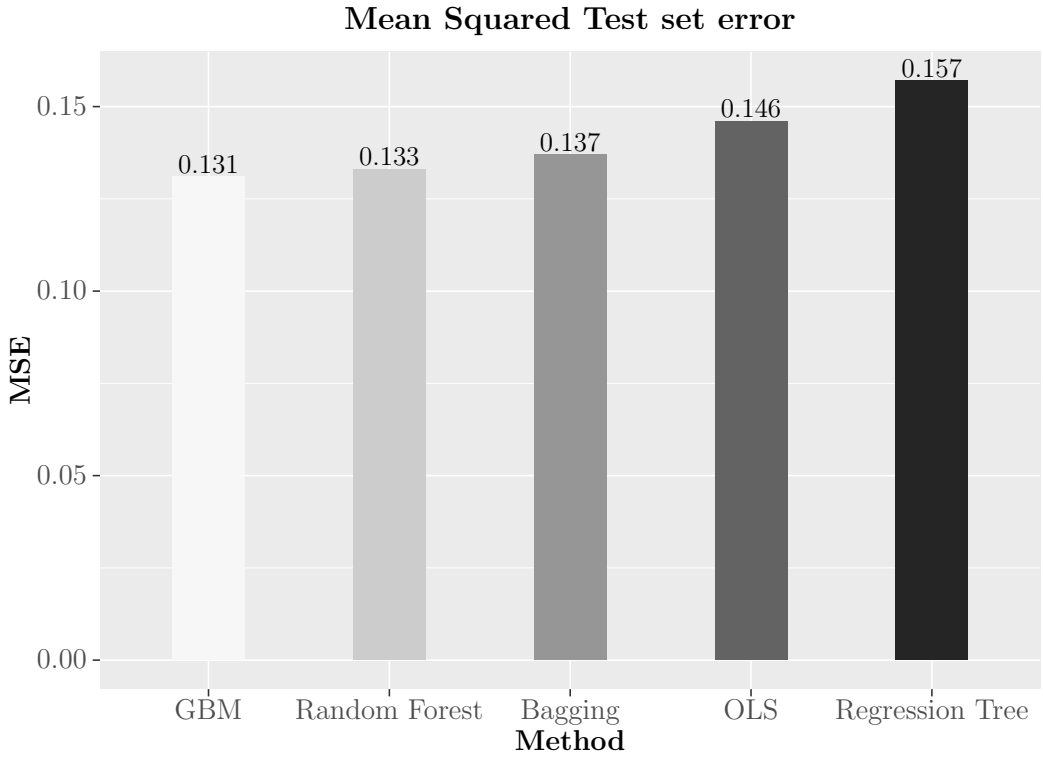


**Figure 4.8:** *MSE for estimated algorithms on test dataset*

The results indicate, that the ensemble methods perform better than the single regression tree. With a score of 0.146 even a linear regression (compare table .2) yields a lower MSE than the regression tree. The prediction of the bagged model leads to a further improvement. Using Random Forest algorithm leads to slight improvement compared to Bagging. Hence, introducing further randomness leads to an improvement w.r..t. the prediction results on the test dataset. GBM yields the lowest MSE on the test data (0.131), yet it is closely followed by the Random Forest.

# 5 Conclusion

This paper analyzes the prediction of gross hourly wages, based on the SOEP data from 2015. For the prediction four different supervised machine learning algorithms are trained on a training dataset to make a prediction on an unseen test dataset. Starting out with a regression tree, subsequently Bagging, Random Forest and Gradient Boosting were applied. The training process includes a grid search to tune the hyperparameters of the corresponding models.
To evaluate which of the algorithms works best, the MSE on the test dataset is calculated for each of them. The results indicate that the ensemble learners deliver better prediction results than the single regression tree. GBM is performing the best, closely followed by the random forest.
In conclusion, this analysis shows that tree based machine algorithms provide a good performance on predicting the gross hourly wages. Especially all ensemble learners perform better than the classical approach of a linear regression. Furthermore the results are very close to each other which underlines the stability of the ensemble learners. An advantage is that the algorithms are also capable to explain and interpret the results by using relative feature importance plots and the partial dependence plots. Occupation, tenure and the size of the firm are the most important features in predicting the gross hourly wages. Yet the predictive performance is better than a linear regression the results of the MSE indicate that there is still room for improvement. In case of gradient boosting machine the grid search for the best set of hyperparameters delivered a corner solution. Although the interval has been chosen quite largely, it looks like the performance is still improvable.
Better results could maybe be obtained by introducing further features, that have not been included in this dataset. Additionaly it would be interesting to see how well the algorithms perform for different survey years.

# 6 Bibliography

# References

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140. 2.2

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. 2.3

Breiman, L., Friedman, J., Stone, C., & Olshen, R. (1984). *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis. 2.1, 2.1

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4), 367 – 378. Nonlinear Methods and Data Mining. 2.4

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org. 1, 3.2.1

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated. 2, 2.1.1, 2.1.1, 2.2, 1, 3.2.1, 3.2.1, 4.2.1, 4.2.2, 4.3

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3), 210–229. 1

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1), 1–47. 1

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 3rd edition. 3.2.1, 3.2.1

# 7 Appendix

# Data

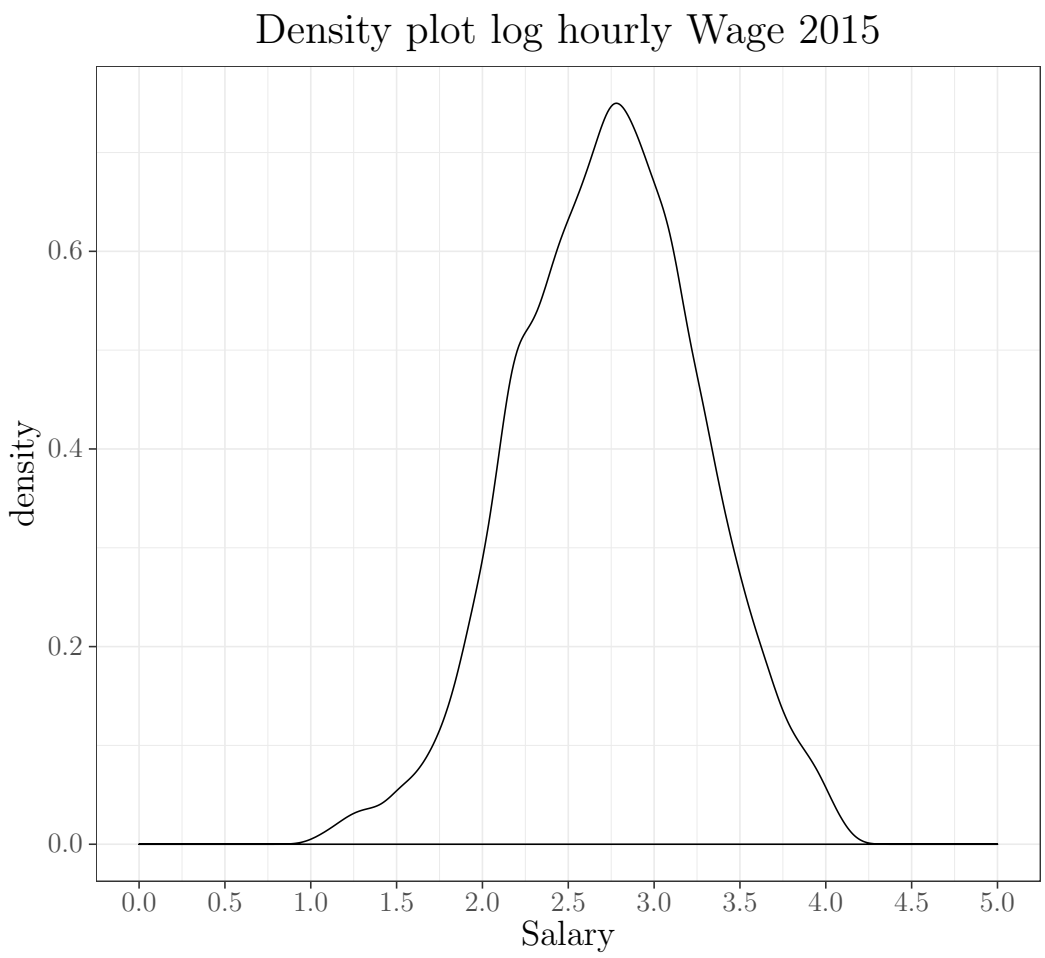


**Figure .1:** *Kernel Density of log(Salary)*

**pgegp** – Erikson and Goldthorpe Class Category

| | | |
|---|---|---|
| 1 | [1] [I] Higher Managerial and Professional Workers | 3287 |
| 2 | [2] [II] Lower Managerial and Professional Workers | 5736 |
| 3 | [3] [IIIa] Routine Clerical Work | 3022 |
| 4 | [4] [IIIb] Routine Service and Sales Work | 3286 |
| 5 | [5] [IVa] Small Self-Employed With Employees | 400 |
| 6 | [6] [IVb] Small Self-Employed Without Employees | 653 |
| 7 | [7] [V] Manual Supervisors | 0 |
| 8 | [8] [VI] Skilled Manual Workers | 3551 |
| 9 | [9] [VIIa] Semi- and Unskilled Manual Workers | 3437 |
| 10 | [10] [VIIb] Agricultural Labour | 330 |
| 11 | [11] [IVc] Self-Employed Farmers | 134 |

**Figure .2:** *Erikson and Golthorpe Class Category*

# Bagging

**Listing 1:** Bagging Output

```
Call:

randomForest(formula = Salary ~ ., data = train,
                          mtry = 12, nodesize = 18,
                          ntree = 500,
                          importance = TRUE)

                Type of random forest: regression
                Number of trees: 500
                No. of variables tried at each split: 12

                Mean of squared residuals: 0.1419925
                      \% Var explained: 49.88
```

# Random Forest

**Listing 2:** Random Forest Output

```
Call:

 randomForest(formula = Salary ~ ., data = train,
                              mtry = 3, nodesize = 12,
                              ntree = 500,
                              importance = TRUE)

                 Type of random forest: regression
                 Number of trees: 500
                 No. of variables tried at each split: 3

               Mean of squared residuals: 0.1387692
                     \% Var explained: 51.02
```

# GBM

**Listing 3:** GBM Output

```
Call:

gbm(formula = Salary ~ ., distribution = "gaussian", data = train,
        n.trees = 10000, interaction.depth = 7, shrinkage = 0.001,
        bag.fraction = 0.5)

A gradient boosted model with gaussian loss function.
10000 iterations were performed.
There were 12 predictors of which 12 had non-zero influence.
```

| Number | Variable | relative influence |
|---|---|---|
| 1 | Occupation | 34.1915063 |
| 2 | Tenure | 14.5296887 |
| 3 | Size | 12.1038573 |
| 4 | Hours | 9.2354786 |
| 5 | Education | 8.8836753 |
| 6 | Male | 5.9684885 |
| 7 | Age | 5.1999632 |
| 8 | West | 3.9272809 |
| 9 | LearnedJob | 3.1522555 |
| 10 | Civil | 1.5953468 |
| 11 | Married | 0.6157677 |
| 12 | Children | 0.5966912 |

**Table .1:** *Feature Importance of Gradient boosting Machine.*

# Linear Regression

| | *Dependent variable:* |
|---|---|
| | Salary |
| Male | 0.1*** |
| | (0.02) |
| Age | 0.003*** |
| | (0.001) |
| West | 0.2*** |
| | (0.02) |
| Married | 0.05*** |
| | (0.02) |
| Children | 0.03*** |
| | (0.01) |
| Hours | 0.01*** |
| | (0.001) |
| Civil | −0.03 |
| | (0.02) |
| Education | 0.04*** |
| | (0.003) |
| Size | 0.1*** |
| | (0.004) |
| LearnedJob | 0.1*** |
| | (0.02) |
| Occupation | −0.1*** |
| | (0.003) |
| Tenure | 0.01*** |
| | (0.001) |
| Constant | 1.3*** |
| | (0.1) |
| Observations | 3,284 |
| $R^2$ | 0.5 |
| Adjusted $R^2$ | 0.5 |
| Residual Std. Error | 0.4 (df = 3271) |
| F Statistic | 261.9*** (df = 12; 3271) |
| *Note:* | *p<0.1; **p<0.05; ***p<0.01 |

**Table .2:** *Linear Regression*