**Simulation Model Testing Memorandum**

Project: Micro-Simulation Model on Worker Leave

Contract #DOLQ129633247

Order #1605DC-17-U-00086

March 15, 2019



**Simulation Model Testing Memorandum**

Project: Micro-Simulation Model on Worker Leave

Contract #DOLQ129633247

Order #1605DC-17-U-00086

March 15, 2019

**Submitted to:**

U.S Department of Labor

Chief Evaluation Office

**Submitted by:**

IMPAQ International, LLC

10420 Little Patuxent Parkway, Suite 300

Columbia, MD 21044

**Table of Contents**

[Executive Summary 2](#_Toc3452899)

[Introduction 3](#_Toc3452900)

[Results 6](#_Toc3452901)

[Bibliography 16](#_Toc3452902)

[Appendix A: Technical Details of Simulation Methods 17](#_Toc3452903)

[Appendix B: Model Parameters for Actual PFL Programs 23](#_Toc3452909)

[Appendix C: Full Description of R Model Testing Results 24](#_Toc3452910)

## Executive Summary

IMPAQ has submitted an alpha version of the Paid Leave Microsimulation model to Department of Labor Chief Evaluation Office in October 2018. Since then, IMPAQ’s model development team have further developed the microsimulation model, by incorporating more model features to better reflect U.S. workers’ leave taking behavior and leave needs, implementing more simulation algorithms to fully leverage the power of machine learning, and testing the model with more extensive data across states and over years. This memo summarizes the finding from our model testing efforts to date for both the R and Python versions of the model. We plan to refine and calibrate our model based on these results.

We have performed three different types of model tests. First, for states that have implemented state-wide paid leave programs, we test how the simulated program costs compare to published program costs. Second, we use solely the FMLA data to perform a k-fold cross-validation, and compare the population-level statistics (such as total number of leave takers) under model prediction against the same statistics estimated from the FMLA data. Third, we compute performance measures of prediction methods by comparing individual-level prediction outcomes and actual individual outcomes observed in the FMLA data. For all three types of model tests, we repeat the test for different simulation methods, including the traditional logit model, various machine learning methods, and a random draw benchmark method.

Our model testing results show that our current model has the capability to simulate program outlays that are close to actual outlays for the three considered states (California, New Jersey, and Rhode Island) and for each state, under certain simulation algorithms. There is however not one single simulation method that outperforms the rest definitively.

We recognize the many discrepancies between the model prediction and observed data through model testing for state program outlays and leave taking and leave needs of the population. These discrepancies may be driven by (i) the choice of predictors in training simulation equations, (ii) the assumption of homogenous take up rates (0.25) across states and across leave types, and (iii) oversimplification of program features for the three state programs. We expect further consolidation of these discrepancies in future model development and testing with more work on optimizing predictor choices, optimizing the use of different simulation methods for different simulation components of the model, reviewing literature on state program take ups, and coding more features of current state programs.

## 1. Introduction

The FMLA dataset offers information on individual workers’ leave taking and leave needs, along with demographic variables. By exploiting the relationship between the demographics and leaves taken and needed, we wish to impute the leave taking and leave needs using a larger worker dataset such as ACS. The ACS has several advantages over the FMLA data set: it contains much richer information of workers and their household, is published more frequently, and can be analyzed for individual states. With this microsimulation framework, it is possible to simulate potential costs of state paid leave programs, given program parameters such as eligibility rules and benefit payout schedules, and the assumption of workers’ take-up rates. The cost simulation can then inform policymakers about whether such a program is viable, and if yes, what the optimal design could be, and how the program cost should be funded. The usefulness of the microsimulation model, therefore, is highly dependent upon whether the model can produce valid estimates from simulation.

Given that our microsimulation model is extensively based on variable imputation, and imputation of variables almost always introduces error that cannot be captured by the imputation equations, we propose to maximize the use of ACS variables in the model to the extent possible, and only simulate leave-taking related variables based on the FMLA data. We identified the following 5 sets of outcome variables that need to be simulated:

1. The 6 binary variables indicating whether individuals take leave of each type (*take\_type* where *type*{*‘own’, ‘matdis’, ‘bond’, ‘illchild’, ‘illspouse’, ‘illparent’*})
2. The 6 binary variables indicating whether individuals need leave of each type (*need\_type* where *type*{*‘own’, ‘matdis’, ‘bond’, ‘illchild’, ‘illspouse’, ‘illparent’*})
3. Whether individuals are financially sensitive to changes in effective cost of leave taking (*resp\_len*)
4. What proportion of pay individuals receive from their employers when on leave (*prop\_pay*)
5. The length of each type of leave taken (*len\_type* where *type*{*‘own’, ‘matdis’, ‘bond’, ‘illchild’, ‘illspouse’, ‘illparent’*})

For this model testing memo, we have completed the coding of the following 6 different methods for imputing these variables into the ACS data.

1. K=5 Nearest Neighbor, based on majority voting (denoted *KNN\_multi* in the model code)
2. K=1 Nearest Neighbors (*KNN1*)
3. Logit Regression (*logit*)
4. Naïve Bayes Classifier (*Naïve\_Bayes*)
5. Random Forest Classifier (*random\_forest*)
6. Ridge Regression Classification (*ridge\_class*)

The details of these simulation methods are provided in Appendix A.

We have implemented a 7th method, random draw imputation (denoted *random*), which will serve as a benchmark method. For each observation in the test data set, the random draw method will simply pick a random observation in the training data set, and assign that training observation’s value to be the testing observation’ value for the imputed variable. During our model testing, all other model parameters are held identical across these 7 imputation methods.

We perform three different types of model tests.

***Comparing simulated and published program costs.*** The ability to closely predict total program cost is arguably the most important feature of a good microsimulation model. There are three states with sufficient historical data on benefit outlays to perform this test on: California, New Jersey, and Rhode Island. For each state, we specified the model parameters so that they can approximate the eligibility rules and benefit payout schedules as closely as possible (see Appendix B for state-specific mode testing parameters). Upon completion of simulation under a given simulation method for a given state, we compute the weighted sum of benefits received by each ACS worker in that state, with weight being the population represented by the worker (i.e. the ACS variable *PWGTP*). This weighted sum is our simulated total program cost and is then compared against the published program outlays of the same state.

***Comparing simulated and observed population level statistics.*** We recognize that the robustness of a microsimulation model cannot be fully verified if we can only confirm that the model can produce good estimates for the final program cost. In addition, we need to validate the model’s capability to approximate the real-world mechanisms by examining a series of key intermediate outputs. In our case, we consider the following intermediate outputs at the population level:

* Total number of leave takers
* Total number of leaves taken
* Total number of leave needers
* Average wage replacement ratio if receiving paid-leave benefit from employer

All of these population level statistics can be computed directly based on the respective variables observed for each worker in the FMLA data, allowing the comparison with the simulated counterparts for model testing purpose. The first two statistics directly affect the program caseload. The third statistic characterizes the size of worker group who would take up the benefit depending upon program generosity. The last statistic is a key parameter in our model that determines the source of benefit paid to the leave takers - in short, the lower replacement ratio from employer would incentivize worker to take up the state program.

***Comparing simulated and observed individual level outcomes.*** We further investigate the validity of our model by tracking its performance at individual level. This is necessary considering that a simulation method can coincidentally make a good prediction for certain population level statistics (e.g. total number of leave takers) while making drastically wrong prediction at individual level. In such cases, the model may produce misleading inferences such as biased results in subgroup analysis, and ultimately misinform policies.

Therefore, we propose to examine how well a model trained from a subset of FMLA data can predict the outcome of workers represented in the rest of the FMLA data. Formally, we perform a k-fold cross validation in which the entire FMLA data is randomly partitioned into equal subsamples. For each fold (subsample) , we use the complement subsample to train the model and make predictions on . We then compute prediction metrics that summarize model performance across all folds. These prediction metrics are then by construction based on performance of the model at individual level.

This distinction between population level and individual level model performance is illustrated inExhibit 1 below. In this example, Method 1 performs better at predicting population levels of leave taking, while Method 2 performs better at predicting who takes leave at an individual level.

Exhibit : Aggregate versus Individual-Level Performance

|  |  |  |
| --- | --- | --- |
| ***Population-Level Performance*** | | |
| *Actual*  Non- Leave taker  2 Leave Takers  Leave  taker | *Predicted - Method 1*  2 Leave Takers – No Error | *Predicted - Method 2*  3 Leaves Takers – 50% overestimate |
| ***Individual-Level Performance*** | | |
| *Actual*  2 Leave Takers | *Predicted - Method 1*  2 Leave Takers – accuracy = 50% | *Predicted - Method 2*  3 Leave Takers – accuracy = 75% |

Note: Accuracy is defined as total number of correct individual predictions (either taker or non-taker) divided by total population.

For this model testing effort, we use the 5-year ACS microdata from 2012 to 2016. We select this ACS period to minimize the time distance from the administration of the FMLA 2012 survey, while allowing collecting as much outlay data as possible from the recently implemented state programs. In the future, we are considering switching to the recently published ACS 2013-2017 data. However, the farther we move the ACS sample period away from the FMLA survey year, the more likely the temporal variation in actual leave taking behavior is to contaminate the observed discrepancy between model estimates and actual state program costs.

## 2. Python Model Results

We discuss our model testing results in this section. For validating the total program costs and population level statistics, we present the 95% confidence interval surrounding the simulated outcome to reflect the underlying variance in sampling. The confidence interval is given by

where is the statistic of interest (e.g. total outlay or total number of leave takers), is the -statistic at confidence interval, and is the standard error of statistic calculated from the 80 replication weights supplied in either ACS or FMLA surveys. Formally, the standard error

where is the statistic computed using replication weight for , and is a adjustment factor to account for any underestimation of the standard error using due to repetitive use of the same sample, although with different weights (Judkins, 1990). For our model testing we follow Census’ practice of setting .

For validating the performance of different simulation methods when simulating individual-level outcomes, we propose to use *accuracy* as the performance measure. Each simulation method in our model can be considered as a classifier (e.g. classifying whether a worker is a leave taker), the *accuracy* of a classifier is defined as

where and respectively denote *positive* and *negative* cases, and and denote *true positive* and *true negative* prediction cases. We choose the *accuracy* measure since we would like to have a good paid leave model that can accurately predict *both* positives and negatives. For example, if the model is poor at flagging *TP*, we would underestimate the number of leave takers and thus the costs of adopting paid leave programs, and states might move imprudently by adopting a very costly program. Likewise, if the model is poor at flagging *TN*, we would overestimate the number of leave takers, and workers might not be offered a program that would otherwise be adopted.

There are other measures such as *precision*, *recall*, *F1-score*, and *specificity* that are commonly used for evaluating performance of classifiers. However, all these measures only consider or *TN* but not both, hence are less useful than *accuracy* for testing our model.

In subsections below, we discuss the model testing results for (i) total program outlay, (ii) population level statistics, and (iii) individual level outcomes.

**2.1 Total Program Benefit Outlays**

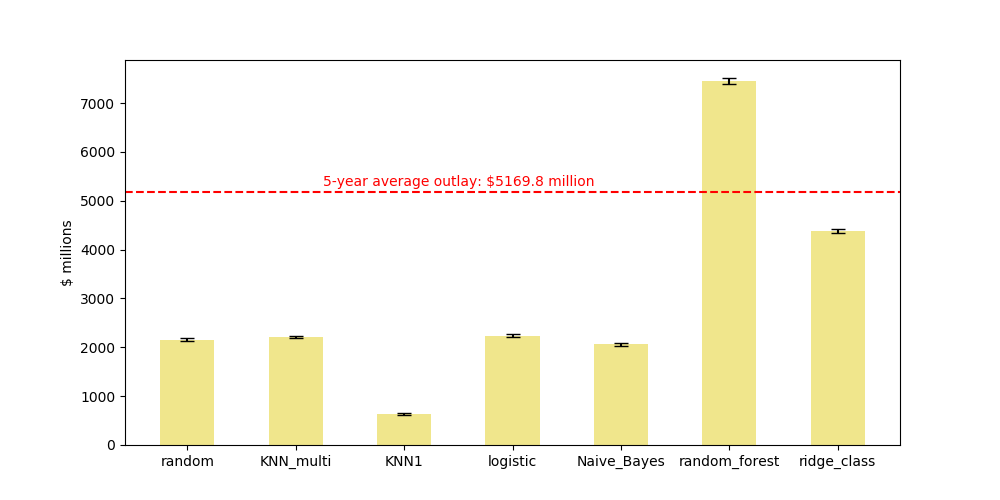
We present the comparison of simulated and actual program outlay in California, New Jersey, and Rhode Island in Exhibit 2. The results show that:

* For New Jersey, 3 of the 6 methods (except the benchmark *random* method) can closely predict the actual program cost, while overestimation occurs for *random\_forest* and *ridge\_class*, and underestimation occurs for *KNN1*.
* For California and Rhode Island, the prediction under *ridge\_class* is the closest to actual cost. Overestimation remains for *random\_forest*, and underestimation prevails across all other methods.
* Across states, the relative magnitude of cost prediction remains similar - simulated outlay is the always the smallest under *KNN1*, larger and fairly similar under *KNN\_multi*, *logistic*, and *Naïve\_Bayes*, and further becomes twice to three times larger under *ridge\_class* and *random\_forest*.

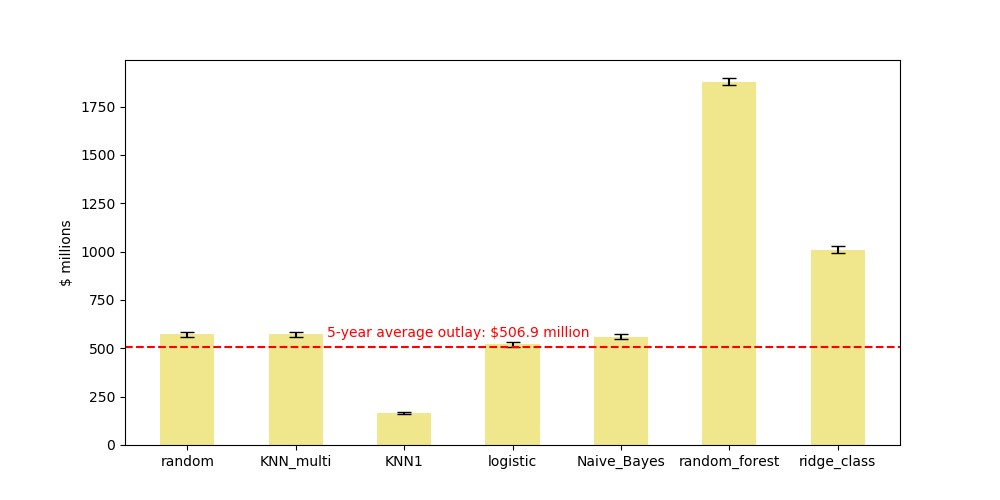
The last observation made above suggests that the bias of each simulation method in predicting program outlay is consistent across applications (for different states and the associated program parameter settings). This motivates us to further investigate for each method how they perform at different stages of simulation in our model, including simulating population level statistics and individual-level outcomes.

Exhibit : Simulated vs. Actual Program Outlay

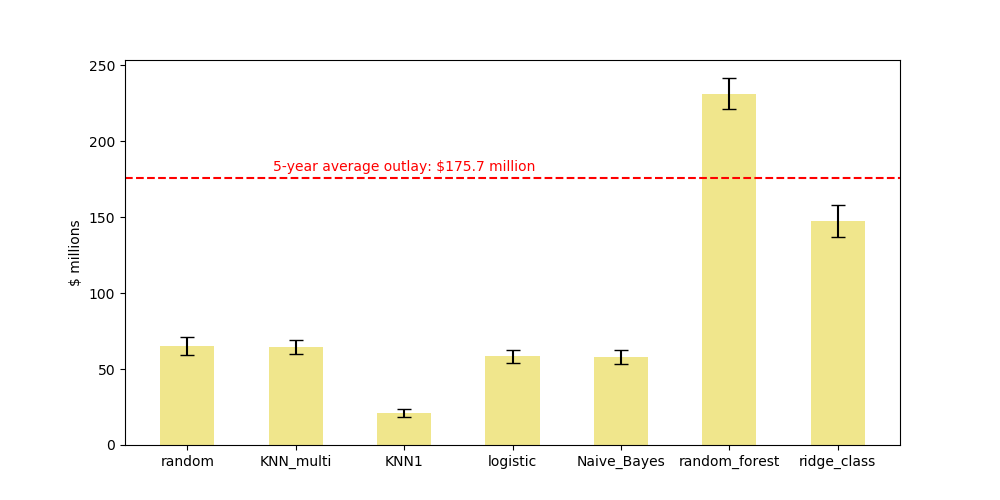
(a) California



(b) New Jersey



(c) Rhode Island



Note: Number of neighbors = 5 in *KNN\_multi*. Confidence intervals are 95%. 5-year average outlay is derived from 2012-2016 published data.

**2.2 Population-Level Statistics**

As discussed in the *Introduction* section, our model testing consider the following four population level statistics that are crucial determinant of total program costs:

* Total number of leave takers
* Total number of leaves taken
* Total number of leave needers
* Average wage replacement ratio if receiving paid-leave benefit from employer

For each statistic, we perform a 4-fold cross-validation using the FMLA dataset, make prediction for the testing subsample in each data fold, and compute the weighted sum and the associated confidence interval. We limit the number of data folds to 4 considering that the FMLA sample size is less than 3,000, thus a 4-fold validation would lead to 700 observations per fold and a training sample size of about 2,100, offering sufficient statistical power for our models.

We present the results in Exhibit 3.The results shows that:

* For all population level statistics, the *random\_forest* method provides predictions that are far above the population level estimate from data. Although higher proportion of pay from employer would discourage workers from taking up the state program, the total number of leave takers and needers as well as number of leaves taken are predicted to be so large under *random\_forest* so that the negative effect of high proportion of pay from employer benefits on total program cost has been completely offset. This therefore leads to the overestimation of total program outlay under *random\_forest* in Exhibit 2.
* The *KNN\_multi* method underestimates all four statistics. The underestimation of the number of leave takers and needers and the number of leaves taken would lead to underestimation of total program cost, while the underestimation of proportion of pay from employer benefits would lead to *overestimation* of total program cost because benefit from employer and benefit from the state program are substitutes under our model assumption that workers cannot simultaneously receive both type of benefits. With the two opposite biases arising from *KNN\_multi*, the overall bias became smaller in magnitude for the total program outlay as shown in Exhibit 2. This is an example that shows the importance of analyzing intermediate model variables besides the final model output - in our case the total outlay, because a fairly ‘good’ prediction of the final outcome can be the joint outcome of two counteracting biased ‘bad’ predictions of intermediate variables.

Exhibit : Cross Validation Results, Population Level Statistics

|  |  |
| --- | --- |
| (a) Total number of leave takers | (b) Total number of leaves taken |
| C:\workfiles\Microsimulation\git\microsim_python\output\figs\old\test_within_fmla_agg_taker.png | C:\workfiles\Microsimulation\git\microsim_python\output\figs\old\test_within_fmla_agg_num_leaves_taken.png |
| (c) Total number of leave needers | (d) Mean proportion of pay from employer benefit |
| C:\workfiles\Microsimulation\git\microsim_python\output\figs\old\test_within_fmla_agg_needer.png | C:\workfiles\Microsimulation\git\microsim_python\output\figs\old\test_within_fmla_agg_prop_pay.png |

Note: Number of neighbors = 5 in *KNN\_multi*. Confidence intervals are 95%. 5-year average outlay is derived from 2012-2016 published data.

**2.3 Individual-Level Statistics**

As shown in Exhibit 1 previously, a well-performing prediction method for population level statistics can perform poorly for predicting individual level outcomes, and vice versa. We therefore devote this subsection to analyzing how different simulation methods can successfully predict outcomes at individual worker level for the entire FMLA workers sample. We continue to use a 4-fold cross validation to maintain sufficient prediction power for our models given the FMLA sample size, and we focus on the *accuracy* performance measure that accounts for both *true positives* and *true negatives* in individual level prediction.

The results are presented in Exhibit 4, suggesting that:

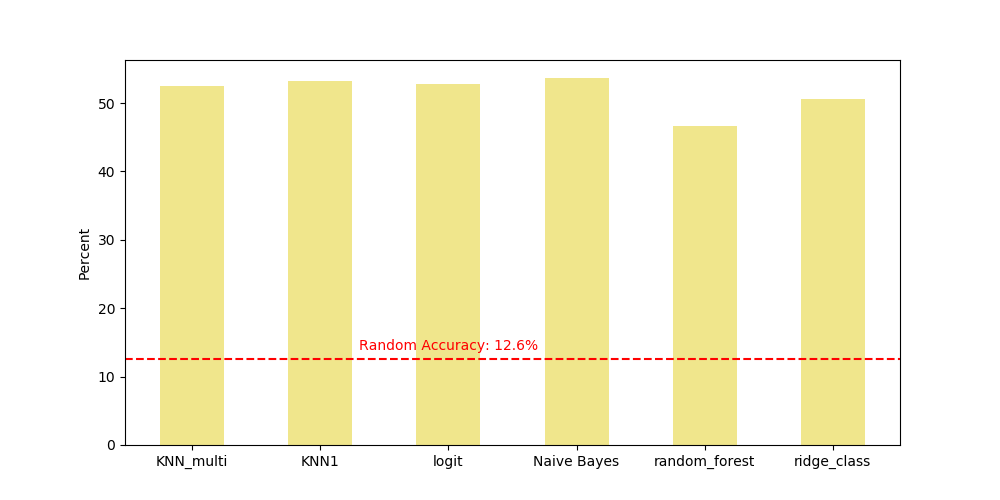
* All methods can perform significantly better than the *random* method (random draws) across all three individual level outcomes, confirming the usefulness of demographic predictors in our model. Traditional methods such as *logit* (logistic regression) can be sometimes outperformed by machine learning methods such as *KNN1* (nearest neighbor) and *Naïve Bayes.*

The *random\_forest* method has particularly low accuracy when predicting leave needing status of individuals. Given the overestimated number of leave needers under this method shown in Exhibit 3, this implies many false positives (false leave needers) flagged under *random\_forest*, and ultimately causing overestimation of total program outlay.

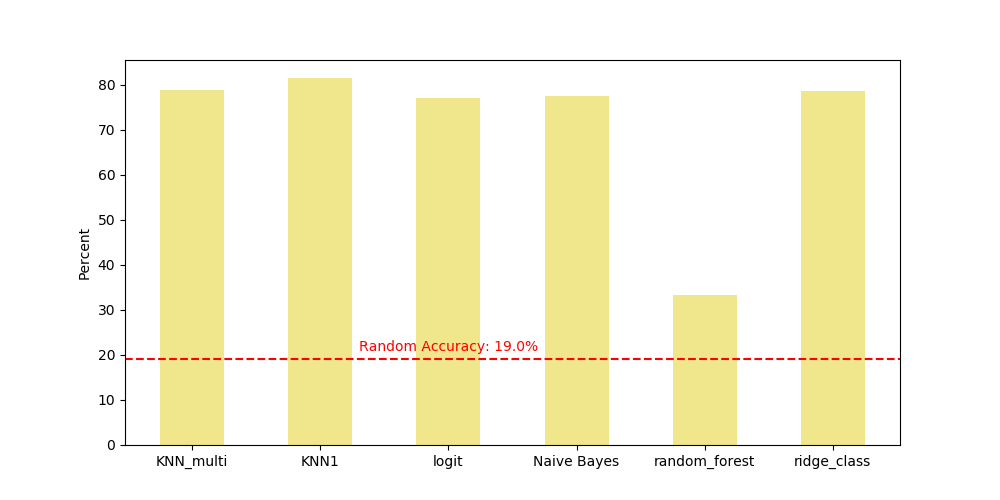
* The prediction accuracy is fairly low for proportion of pay from employer, which is characterized as 6 categories in the FMLA data thus leads to a 6-category multinomial classification problem that is more challenging than the other binary ones. The accuracy under *random* is essentially zero because the random draws are made from the entire FMLA sample among which many workers do not report proportion of pay from employer. We are able to achieve an accuracy ranging from 4% to 8% in model testing since we restricted the sample to those who reported positive pay from employer.

Exhibit : Cross Validation Results, Individual Level Outcomes

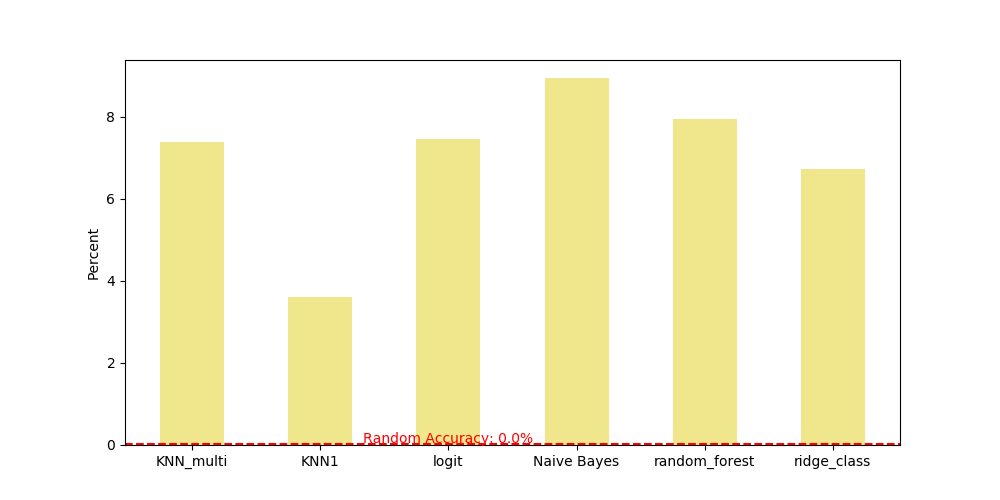
(a) Prediction accuracy, whether a worker is a leave taker



(b) Prediction accuracy, whether a worker is a leave needer



(c) Prediction accuracy, for 6 categories of proportion of pay from employer benefits



Note: Number of neighbors = 5 in *KNN\_multi*. Confidence intervals are 95%. 5-year average outlay is derived from 2012-2016 published data.

**2.4 Model Testing in R**

Besides the above testing effort implemented in Python, we have also performed model testing in R. The R model’s testing results are similar in structure to the Python results, and we present a brief summary of them here. A full discussion of the R model’s results is included in Appendix C.

Until this stage of the project, our Python and R model developers have been working fairly independently on the two versions of the model in order to effectively validate each other’s work. We aim to consolidate two versions of the model as we approach the later stage of this project, and we have taken a few early steps including (i) validating the data pre-processing code to ensure that all resulting cleaned datasets are identical in Python and R models, and (ii) consolidating the model structure between Python and R versions, including the sequential order of simulating different variables if any sample conditioning is involved. For this model testing work, the Python and R models have not been entirely consolidated, thus many discrepancies still exist between them.

We include our model testing results from R version of the model in this section as a reference, and present them in Exhibit 5 through 7 below. Overall, the R results are different from Python ones, including an unexpected level of overestimation under *Naïve Bayes* across all model outcomes and statistics. The resolution of such model testing anomalies will be among our next steps, and we expect many of the differences between Python and R results to consolidate as our model developers continue to validate each other’s work.

Exhibit : Simulated vs. Actual Program Outlays, R

|  |  |
| --- | --- |
| (a) | (b) |
| C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\12_CA_bene_outlay_tot.png | C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\11_NJ_bene_outlay_tot.png |
| (c) | |
| **C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\10_RI_bene_outlay_tot.png** | |

Exhibit : Population Level Statistics, R

|  |  |
| --- | --- |
| (a) | (b) |
| C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\6_leave_takers_tot.png | C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\4_num_leaves_tot.png |
| (c) | (d) |
| C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\6a_leave_needers_tot.png | C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\1_prop_pay_avg.png |

Exhibit : Individual Level Outcomes, R

|  |  |
| --- | --- |
| (a) | (b) |
| C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\7_leave_takers_acc.png | C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\7a_leave_needers_acc.png |
| (c) | |
| ***C:\Users\lpatterson\AnacondaProjects\microsim_R\exhibits\2_prop_pay_acc.png*** | |

**2.5 Conclusion and Next Steps**

Since submission of the alpha version of our model, we have made progresses on our model development by incorporating more model features to better reflect U.S. workers’ leave taking behavior and leave needs, implementing more simulation algorithms to fully leverage the power of machine learning, and testing the model with more extensive data across states and over years. We therefore have a microsimulation model that is sufficiently functional to allow us perform the above model testing. We found that our current model has the capability to simulate program outlays that are close to actual outlays for certain states (California, New Jersey, and Rhode Island), and under certain simulation algorithms. There is however not one single simulation method that outperforms the rest.

We recognize the many discrepancies between the model prediction and observed data through model testing for state program outlays and leave taking and leave needs of the population. We identified the following factors that may drive these discrepancies:

* Choice of predictors in training simulation equations
* Application of a single simulation method throughout different components of the model
* Assumption of homogenous take up rates (0.25) across states and across leave types
* Oversimplification of program features for the three state programs

To address these issues, we will further validate the model to optimize the predictor choices, optimize the use of different simulation methods for different simulation components of the model, scan the literature for evidence of program-specific take up rates in different states, and incorporate model features to better represent current state programs. We expect these efforts to improve the results of our next round of model testing. We also expect to identify and resolve additional issues in both Python and R models when our developers continue to move from independent coding to collaboration. Last but not least, we will work on optimizing the execution speed of the model such as leveraging vectorization in Python, in order to facilitate further model testing and improve user experience.

## 3. Bibliography

Judkins, D. R. (1990). Fay’s method for variance estimation. *Journal of Official Statistics*, *6*(3), 223-239.

Saunders, C., Gammerman, A., & Vovk, V. (1998). Ridge regression learning algorithm in dual variables.

## Appendix A: Technical Details of Simulation Methods

### A.1. Nearest Neighbor

Nearest Neighbor (NN) methods are based on the simple idea that “nearby” data points tend to be similar to each other. The methods work by finding the datapoints that are closest to a given observation in -dimensional space (where is the number of features or predictive variables used) and then tallying the classes of those neighbors. The data point is then predicted to belong to the class that appears the most among its neighbors.

The different variants of NN methods differ according to how they define the measure of “distance”, which is in turn used to define what points are “nearby”. Exhibit 5 presents a selection of popular distance metrics commonly used in NN methods. Note that each metric has its own tunable parameters, and some can be viewed as special cases of another. For example, the Minkowski distance is equal to the Manhattan distance if and the Euclidean distance if .

Exhibit 8: Alternative Distance Metrics for Nearest Neighbor Methods

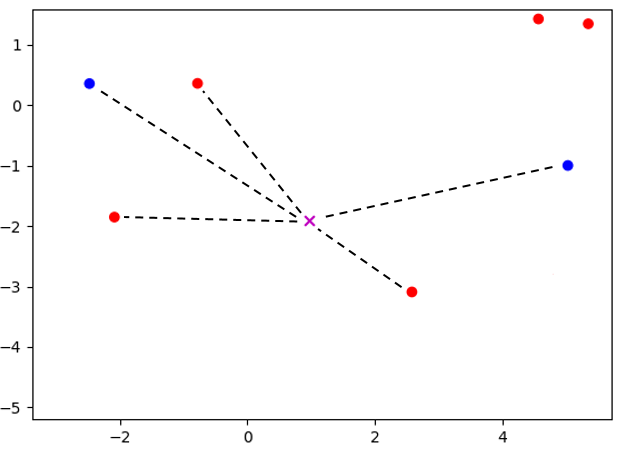
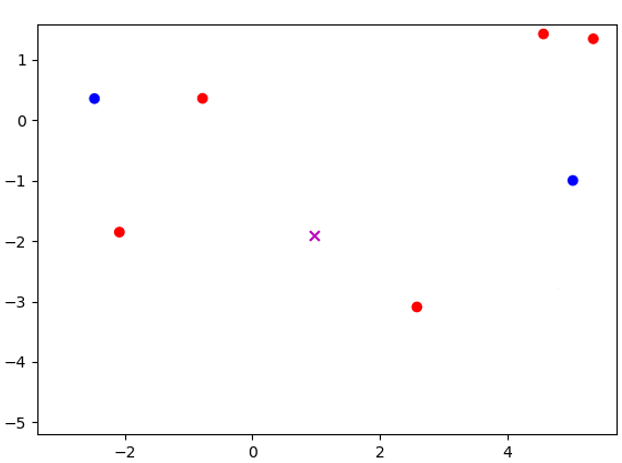
|  |  |
| --- | --- |
| Distance Metric | Description |
| Euclidean |  |
| Manhattan |  |
| Minkowski |  |
| Chebyshev |  |

We now turn to an example that considers three pieces of information about a group of people - height in inches, weight in pounds, and age in years – so that each person is represented in 3-dimensional space with the vector *(height, weight, age)*. The Euclidean distance between individuals **A** = (74, 180, 17) and **B** = (66, 115, 33) is:

Because all the metrics treat the distance between each feature in an identical fashion, NN algorithms perform better when all data features have the same scale. Normalizing each of the feature variables in the pre-processing step is therefore important before training the classifier.

Another example is presented in Exhibit 11. The left-hand panel shows a scatter plot representing a set of training data in 2-dimensional space, with the color of each point representing that point’s class (red or blue). The purple *x* represents a new, unclassified point within the test data. The right-hand panel illustrates how the NN classifier finds the *k=5* closest points using the Euclidean distance. Of these five nearest neighbors, three are red and two are blue.

Exhibit 9: Alternative Distance Metrics for Nearest Neighbor Methods



NN classifiers are intuitive, easy to implement, and quick to train because they only need to store the training data. However, prediction can be computationally intensive because of the need to calculate the distance between the new data point and each point in the training data. If the training data is especially large, with many features, the NN algorithm might not be a viable machine learning option without first employing dimensionality reduction techniques such as PCA and feature selection.

Finally, the number of neighbors will affect the bias and variance of nearest neighbor classifiers. If *k* is too low, the classifier will have a high variance because a prediction will be decided by a small number of neighbors. This flaw might not show up during testing, but will be apparent once the classifier is used with new data. On the other hand, a high *k* value will increase bias because the neighbor groups will become too large, thereby incorporating points that should not be included. Choosing the right value for is important for improving a classifier’s performance.

### A.2. Logistic Regression

**Logistic regression** models the log-odds of the occurrence of an event (in this case participation status), i.e. where is the probability of occurrence. The full model is

where represents the predictable part of the latent log-odds , and is a random error term with logistic distribution that represents the unpredictable part of . In the data, we observe the binary outcome , the actual participation status, instead of the latent log-odds which is a function of , the unobserved *ex ante* probability to participate. The log-odds converts the probability from its domain to the *logit* term with its domain over the entire real line. Upon finding the optimizing , the predicted *probability* of event occurrence is

for each observation in the sample. Since the domain of the exponential term in the denominator is , the predicted probability will always stay within the theoretical interval. The model then can freely optimize for parameter without the issue of making out-of-domain predictions, which would often occur with linear probability models that directly model and predict the probability term .

### A.3. Naïve Bayes

Naïve Bayes methods are among the oldest, simplest and most popular machine learning classifications techniques. Their popularity stems from being extremely fast and surprisingly accurate given the strong assumptions. Because of these properties they are often employed as a baseline method for classification problems.

Naïve Bayes methods refer to a family of classifiers based on applying Bayes' theorem with strong independence assumptions between the features. In our classification problem, the probability of participating for a given set of physician-practice “features” is denoted and can be written using Bayes formula as follows

Deciding on the appropriate value of y is based on the quantity , which is defined as follows

If we guess that the the features X come from a physician that did submit MIPS data (. We can easily estimate the quantities and by taking the respective proportion of each outcome in the training data set, which are about 0.85 and 0.15, respectively.

The central challenge of Naïve Bayes algorithms is therefore to estimate the two functions . If X contains multiple variables, then represents a joint distribution which can require extremely large datasets and significant computational resources to estimate. Naïve Bayes algorithms drastically simplify this estimation process by assuming that each of the features are independent and therefore

so we can estimate the conditional probability distribution of each feature separately. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

The many variations of Naïve Bayes algorithms differ according to how they model the unknown functions. The simplest method, which is used when the feature variable is numerical, is to assume that the conditional probability follows a Gaussian distribution so the estimation procedure simply involves estimating the mean and variance of the data for the class and the class separately. In the quintessential Naïve Bayes application of text classification, it is assumed that the features represent counts, which are modelled as multinomial distributions.

Because our dataset contains both categorical and numerical data, as detailed in Exhibit 1, we trained both the Multinomial method on the categorical data and the Gaussian method on the continuous data. The Multinomial approach significantly outperformed the Gaussian approach (which is perhaps unsurprising given that most variables are categorical) so we report the results only for the Multinomial approach.

### A.4. Random Forest

We first introduce the *decision tree classifier*, which is the basis of the random forest method.

A decision tree classifier is a graph of nodes where each node is associated with a data feature and a set of values related to that data feature. When an observation is passed through the tree, a “decision” is reached at each node based on its feature and associated values, which determines the child node the data point is passed onto. For example, in the case when there are only two child nodes, the algorithm compares the value of the specified feature of the sample data with the node’s value. If the sample’s value is less, the algorithm continues to the node’s left child. Otherwise, it goes to the right child. This process continues at each node until a leaf is reached, which contains a class label that is used to classify the data sample.

A simple decision tree classifier is trained by testing every possible feature and value combination on each node. A common approach to this is through the use of the *Gini Impurity* (GI)value, which measures the quality of a binary separation induced by a node. For data set X with samples and classes, the GI of the set is

where is the percentage of samples belonging to class in the dataset. A lower GI value means that the samples in the left and right set mostly belong to one class. A value of 0 means that both sets are pure, containing only one unique class. The algorithm finds the best possible feature and value combination to minimize the GI value of the current data set. The algorithm then recursively performs the same operation on each child node until a set consists of samples of only one class, which is then used to form a leaf node.

Exhibit 9 presents an example of a decision tree classifier in action on a two-feature dataset of height *h*, measured in inches, and weight *w*,measured in pounds. The algorithm first divides that data according to whether the value of *h* is greater than or less than 69. The GI of this separation can then be calculated as follows:

It turns out that 2/9 is the best attainable GI for this data set, so a node is therefore created with *feature* = and *value* = 69.

Exhibit 10: Decision Tree Classifier Example

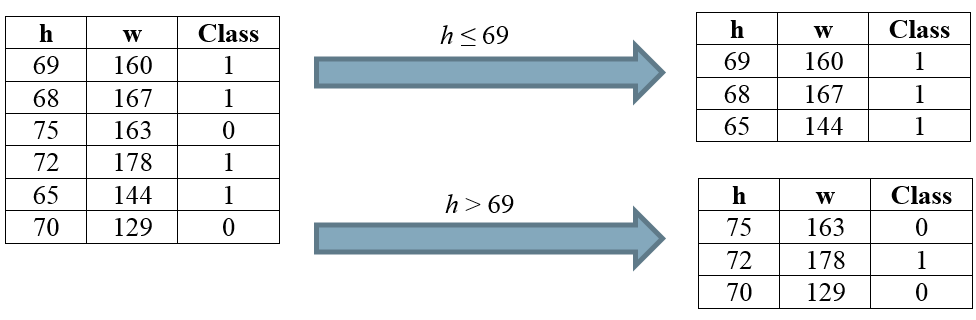
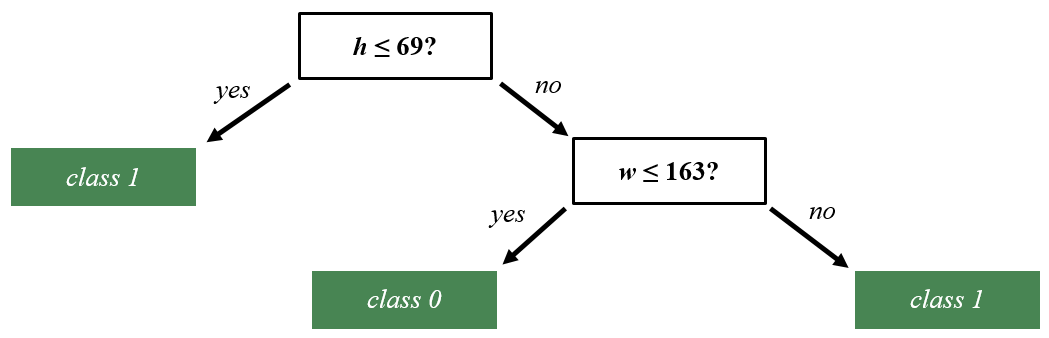


Exhibit 10 presents the optimal decision tree for the data presented in Exhibit 9. Because the data with all belong to class 1, the left-hand child of the parent node is a leaf. For the right-hand node, the classes can be divided using *feature* = and *value* = 163, which completes the decision tree. Consider an unclassified sample with features *h* = 71, *w* = 152. The decision tree predicts that this sample has *class\_value* = 0.

Exhibit 11: Decision Tree Classifier Example



A *Random Forest* classifier is an ensemble-learning model which aggregates multiple “weak learning” methods to create a strong classifier. The weak learning methods are usually simple decision trees. Random Forest methods create many individual decision trees with different subsets of the training data, and where each node of each decision tree is split using a randomly selected attribute from the data. Each tree in the forest predicts the class of a new data sample, and the class that is predicted by the most trees is ultimately predicted by the forest. Random forest methods have increased bias but also decreased variance. Overall, they tend to see an improvement in accuracy over simple decision trees.

### A.5. Kernel Ridge Regression

Ridge regression methods are similar to traditional least-square methods but where the estimated coefficients are “shrunk” toward zero. More formally, the estimated coefficients are defined as follows

where the first term in the expression is the standard loss function in a least-squares problem and the second term is a penalty function with a tuning parameter that controls the strength of the penalty.

Ridge regression was initially introduced as a technique for analyzing multiple regression data that suffer from multicollinearity (as is common in many machine learning applications). When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors in the hope that the net effect will be to give estimates that are more reliable.

A Kernel Ridge Regression is simply a Ridge Regression where the feature variables are first converted into a higher-dimensional object so that the optimization problem becomes

where is a kernel that can be thought of as an inner product of combinations of features. Without going into the detail behind the mechanics, this formulation allows for a Richard feature space while also maintaining efficient computation.[[1]](#footnote-1)

## Appendix B: Model Parameters for Actual PFL Programs

The table below presents the list of parameters that were altered from state to state to conduct simulations of their actual state paid leave programs. These are based on the real-world eligibility requirements for these state’s programs

| **Parameter** | **Description** | **California** | **New Jersey** | **Rhode Island** |
| --- | --- | --- | --- | --- |
| bene level | Proportion of pay received as part of program participation (also known as the wage replacement rate) | 0.55 | 0.66 | 0.6 |
| maxlen own | Max number of days **own health** benefits can be claimed within a 12 month period | 260 | 130 | 150 |
| maxlen illspouse | Max number of days **ill spouse** benefits can be claimed within a 12 month period | 30 | 30 | 20 |
| maxlen illchild | Max number of days ill **child** benefits can be claimed within a 12 month period | 30 | 30 | 20 |
| maxlen illparent | Max number of days ill **parent benefits** can be claimed within a 12 month period | 30 | 30 | 20 |
| maxlen matdis | Max number of days **maternal disability** benefits can be claimed within a 12 month period | 260 | 130 | 150 |
| maxlen bond | Max number of days **child bonding** benefits can be claimed within a 12 month period | 30 | 30 | 20 |
| maxlen DI | Max number of days **maternal disability** **and own health benefits combined** can be claimed within a 12 month period | 260 | 130 | 150 |
| maxlen PFL | Max number of days **child bonding and ill child/parent/spouse benefits combined** can be claimed within a 12 month period | 30 | 30 | 20 |
| maxlen total | Max number of days **benefits of all types combined** can be claimed within a 12 month period | 260 | 130 | 150 |
| week bene cap | Max weekly benefits that can be collected as an absolute value | 1216 | 594 | None |
| week bene cap prop | Max weekly benefits that can be collected as a proportion of the state’s mean weekly wage | None | None | 0.85 |
| earnings | Eligibility Requirement - Minimum earnings (in dollars) within past 12 months | 300 | 8400 | 11520 |
| Own elig adj | Program eligibility adjustment factor for **own health** leave | 1 | .7 | 1 |
| Matdis elig adj | Program eligibility adjustment factor for **maternity disability** leave | 1 | .7 | 1 |

## Appendix C: Full Description of R Model Testing Results

This appendix describes in more detail the results from our first round of validation testing for the R version of the Paid Leave Microsimulation model.

**Preliminary Conclusions**

* **KNN1 is the strongest performing method so far.** We plan to use KNN1 as the default imputation method for the model should these results hold through further testing.
* **No model’s individual-level performance is consistently better than random draws.** Rankings of individual-level performance amongst methods are not consistent either.
* **Better individual-level predictive performance tends to not be related to overall population-level performance.** Given our research question ultimately care most about an accurate population-level performance (and the extent to which individual-level performance affects population-level performance on other data sets is unknown), we place more weight on the methods that perform best with overall population-level predictions. This is validated by a comparison to actual benefit outlay results; models tend to perform similarly to their FMLA population-level performance results.
* **Together, this suggests the limiting of the use of the microsimulation model to answer research questions with aggregate-level results**. For example, the model would be more confident in answering “how many individuals will take leave in State X?” than “which individuals will take leave in State X”.

Below, Exhibit 12 is a summary of the comparative scores for each test that we conducted on the model. It is an approximate measure of how well each method did comparatively to other methods on each measure. While the weight to put on each performance measure is not necessarily equivalent, we see this table as a good visual aid to generally compare method performance across all the different measures we tested. We provide further discussion and speculation on what measures should be considered more or less important in the remainder of the section.

Broadly speaking, the more “Green” boxes in each column, the better the column’s method has performed. Conversely, the more “Red” boxes, the worse the performance. From the general patterns observed in this exhibit, our preliminary conclusions are:

* KNN1 is the best-performing method,
* Naïve Bayes is the worst-performing method,
* Ridge Class, Random Forest, KNN Multi and Logit are somewhere in the middle of those two.

**Exhibit 12.**

**Method Performance Summary**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exhibit # | Measure | Measure Type | Random | Logit | KNN1 | KNN Multi | Random Forest | Naïve Bayes | Ridge Class |
| 13 | Benefits Outlaid – RI | FMLA-to-ACS |  |  |  |  |  |  |  |
| 14 | Benefits Outlaid – NJ | FMLA-to-ACS |  |  |  |  |  |  |  |
| 15 | Benefits Outlaid – CA | FMLA-to-ACS |  |  |  |  |  |  |  |
| 16 | Predicted/Actual Leave Takers | FMLA-to-FMLA Aggregate | N/A |  |  |  |  |  |  |
| 17 | Predicted/Actual Number of Leaves Taken | FMLA-to-FMLA Aggregate | N/A |  |  |  |  |  |  |
| 18 | Predicted/Actual Prop Pay | FMLA-to-FMLA Aggregate | N/A |  |  |  |  |  |  |
| 19 | Predicted/Actual Leave Needers | FMLA-to-FMLA Aggregate | N/A |  |  |  |  |  |  |
| 20 | Leave Takers Accuracy | FMLA-to-FMLA Individual |  |  |  |  |  |  |  |
| 21 | Prop Pay Accuracy | FMLA-to-FMLA Individual |  |  |  |  |  |  |  |
| 22 | Leave Needers Accuracy | FMLA-to-FMLA Individual |  |  |  |  |  |  |  |

**Legend**

Poor Performance Good Performance



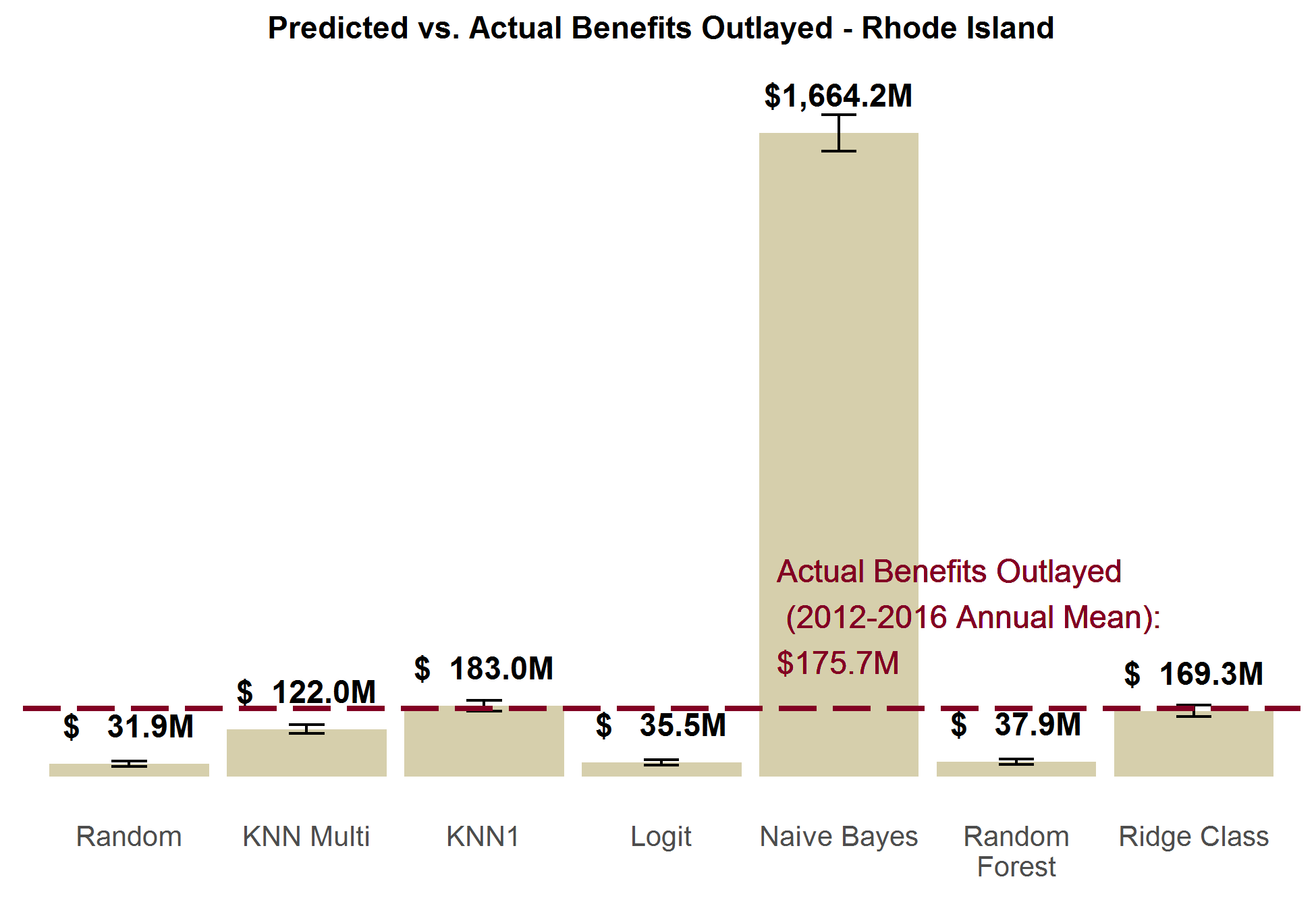
### In the next three subsections, we present and discuss the results of each of the three measure types (FMLA-to-ACS, FMLA-to-FMLA Aggregate, and FMLA-to-FMLA Individual).

### C.1 Benefit Outlays

There are three states with sufficient historical data on benefit outlays to perform this test on: California, Rhode Island, and New Jersey. Results from these simulations are shown below.

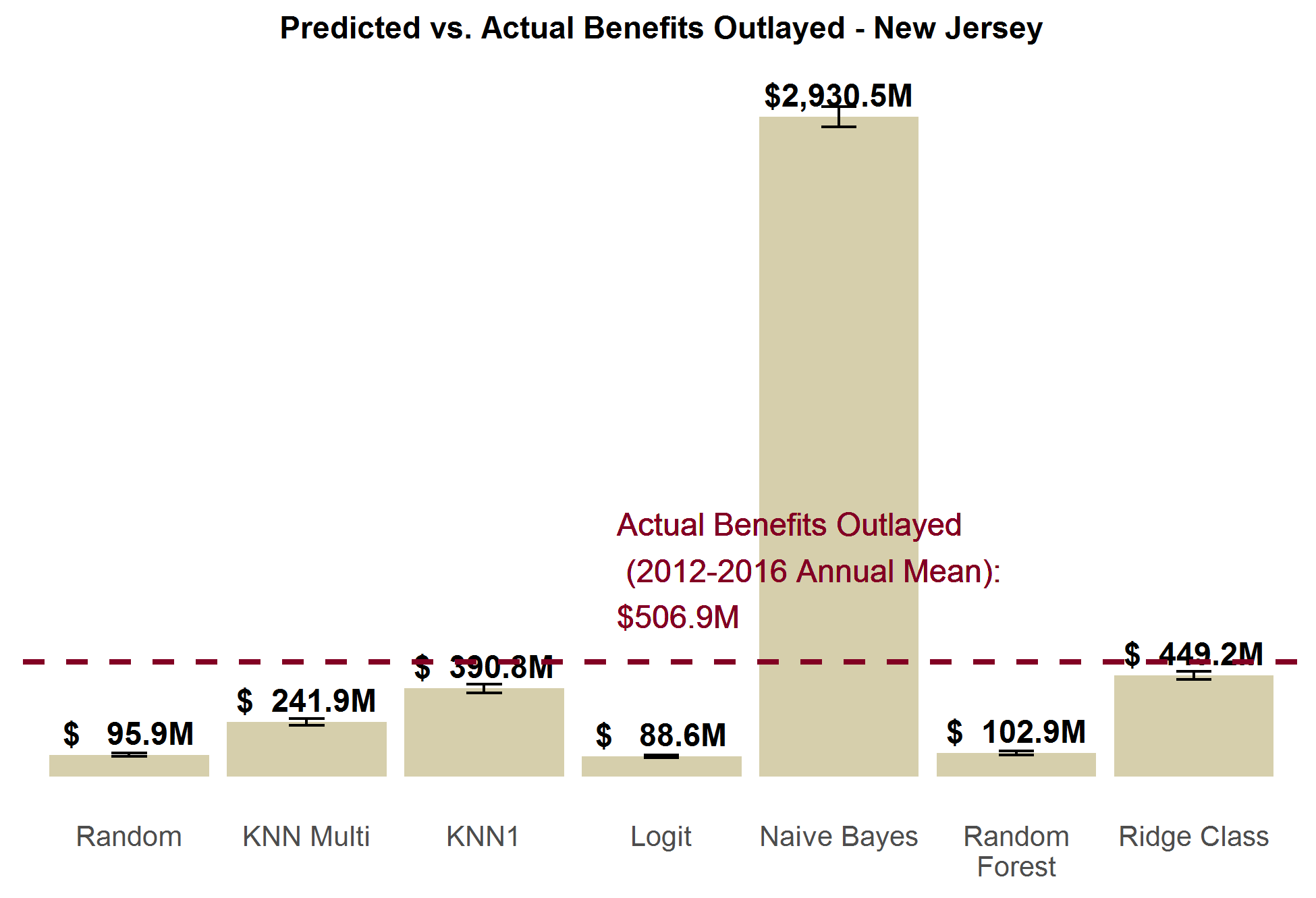
***Rhode Island.*** In Rhode Island, the actual program paid out approximately $176 million in benefits annually between 2012 and 2016. As seen in Exhibit 13 below, two of the prediction methods (KNN1 and ridge class) successfully capture the actual value within their 95% estimate confidence interval. KNN multi performs next best, but still under-predicts benefits by $53 million. Logit and random forest methods even more drastically under-predict benefit outlays, only estimating approximately $36 million and $38 million respectively. These results are similar to random draws, which we yield an estimate of $32 million. Naïve Bayes exorbitantly over-predicts benefit payouts by almost ten-fold, with an estimate of $1.6 billion.

**Exhibit 13.**

****

***New Jersey.*** New Jersey paid out an average of $507 million in benefits annually from 2012 to 2016. We see similar patterns in the predicted benefits outlaid as in Rhode Island in Exhibit 14. KNN1 and ridge class come closest to correctly predicting benefits. But in the case of New Jersey, both methods underestimate benefits by a statistically significant margin. Similarly, logit, random forest, and KNN multi methods all more drastically undershoot benefit estimates, while Naïve Bayes drastically overshoots it.

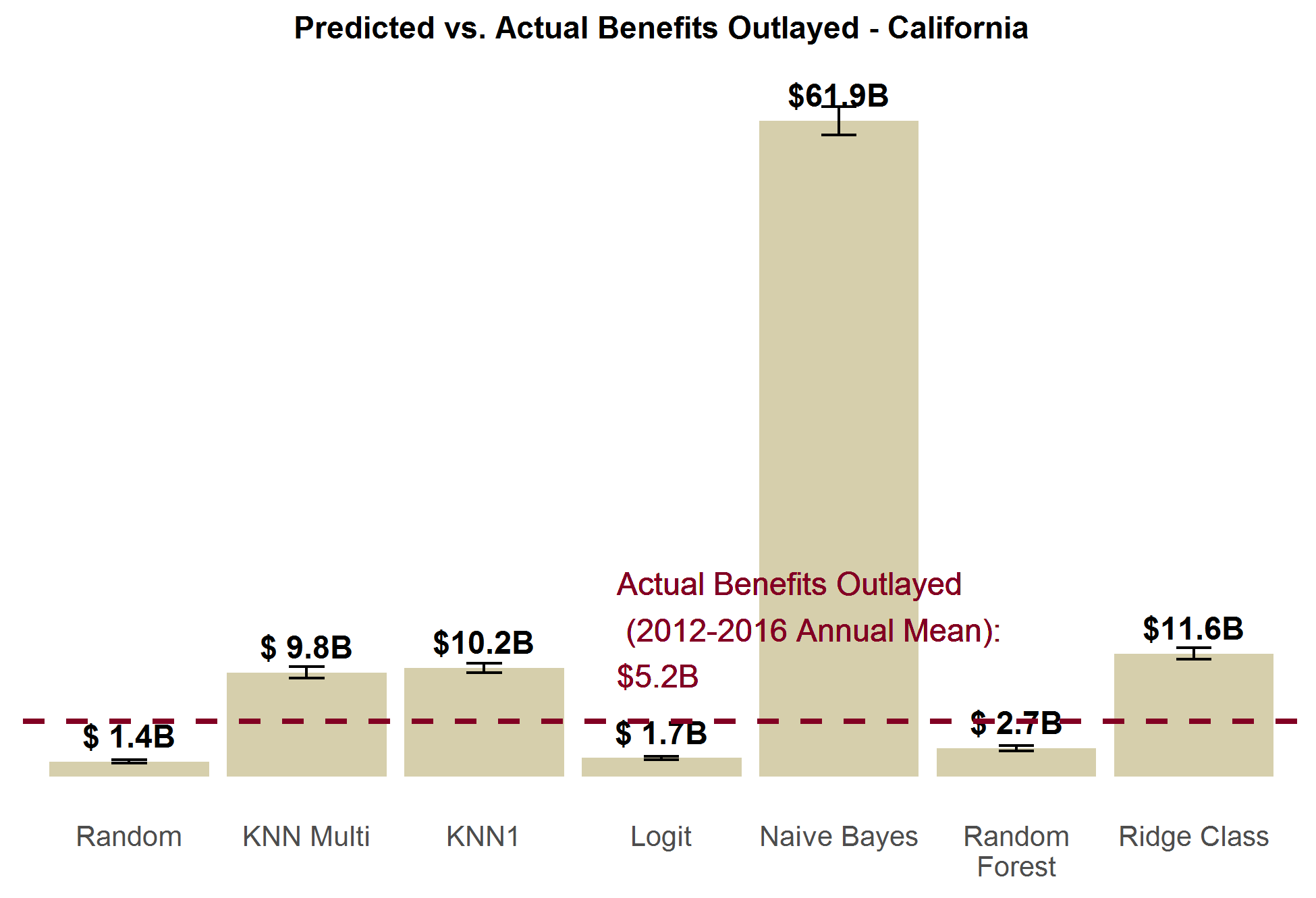
**Exhibit 14.**



***California.*** California’s results are given in Exhibit 15 below. Of the current models, random forest comes closest to estimating California’s actual benefit outlay, though it still arrives at an estimate that about half of the actual outlays ($2.7 billion vs. $5.2 billion). Conversely, KNN multi, KNN1, and ridge class all overestimate benefits outlays by twice the actual benefits paid out. Like the in other states, Naïve Bayes massively overstates the benefits by over tenfold, and the random / logit methods drastically understate the benefit outlays.

We are investigating reasons why California’s predictions are somewhat worse than the other states overall.

**Exhibit 15.**

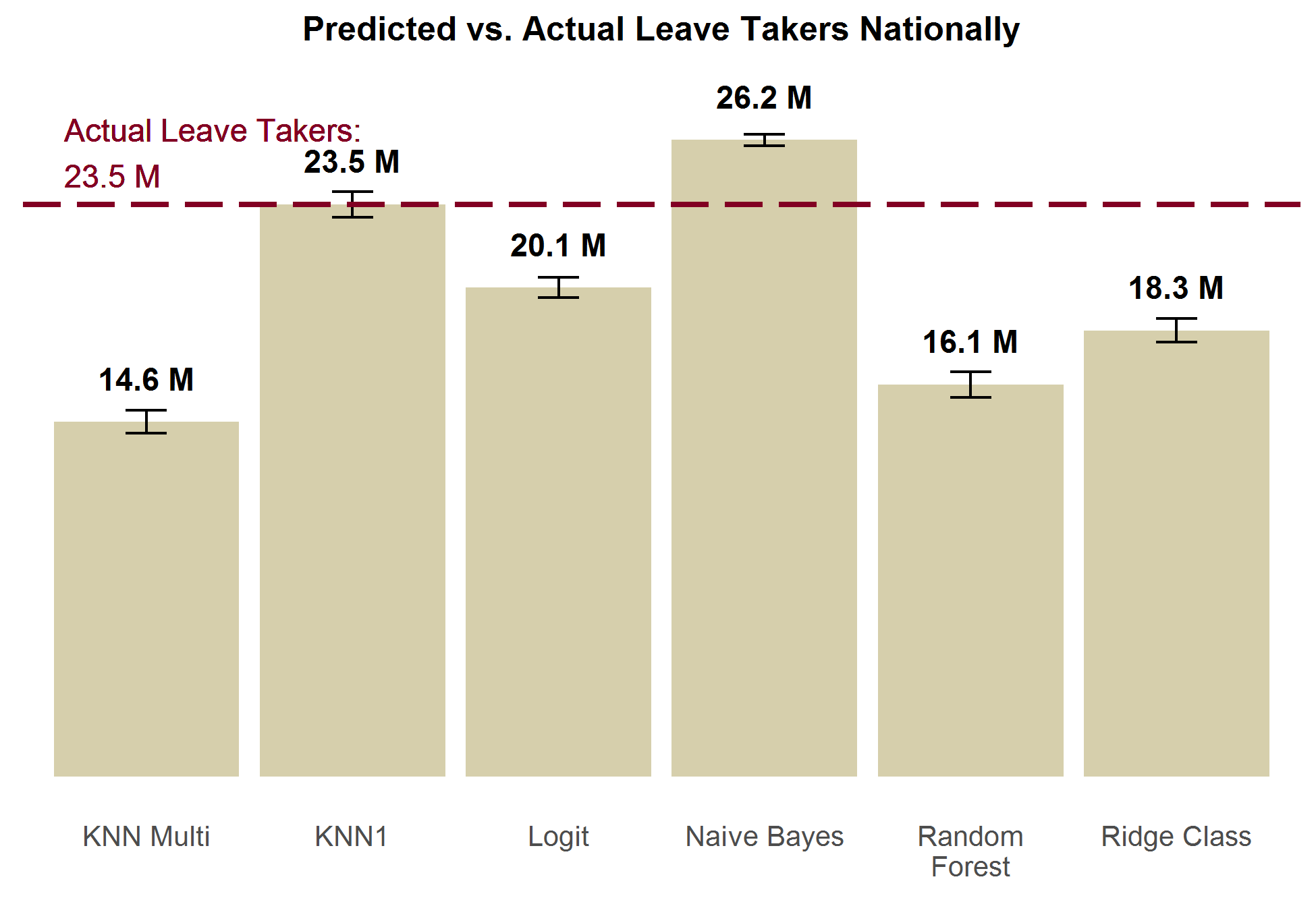


### C.2 FMLA-to-FMLA Performance in Aggregate

When we use part of the FMLA data to make predictions on the other part of FMLA, we can see how well the methods perform at predicting population levels of leave taking. We test the aggregate performance of a number of different variables imputed by the model: number of leaver takers, number of leaves taken, proportion of pay received from the employer, and number of leave needers.

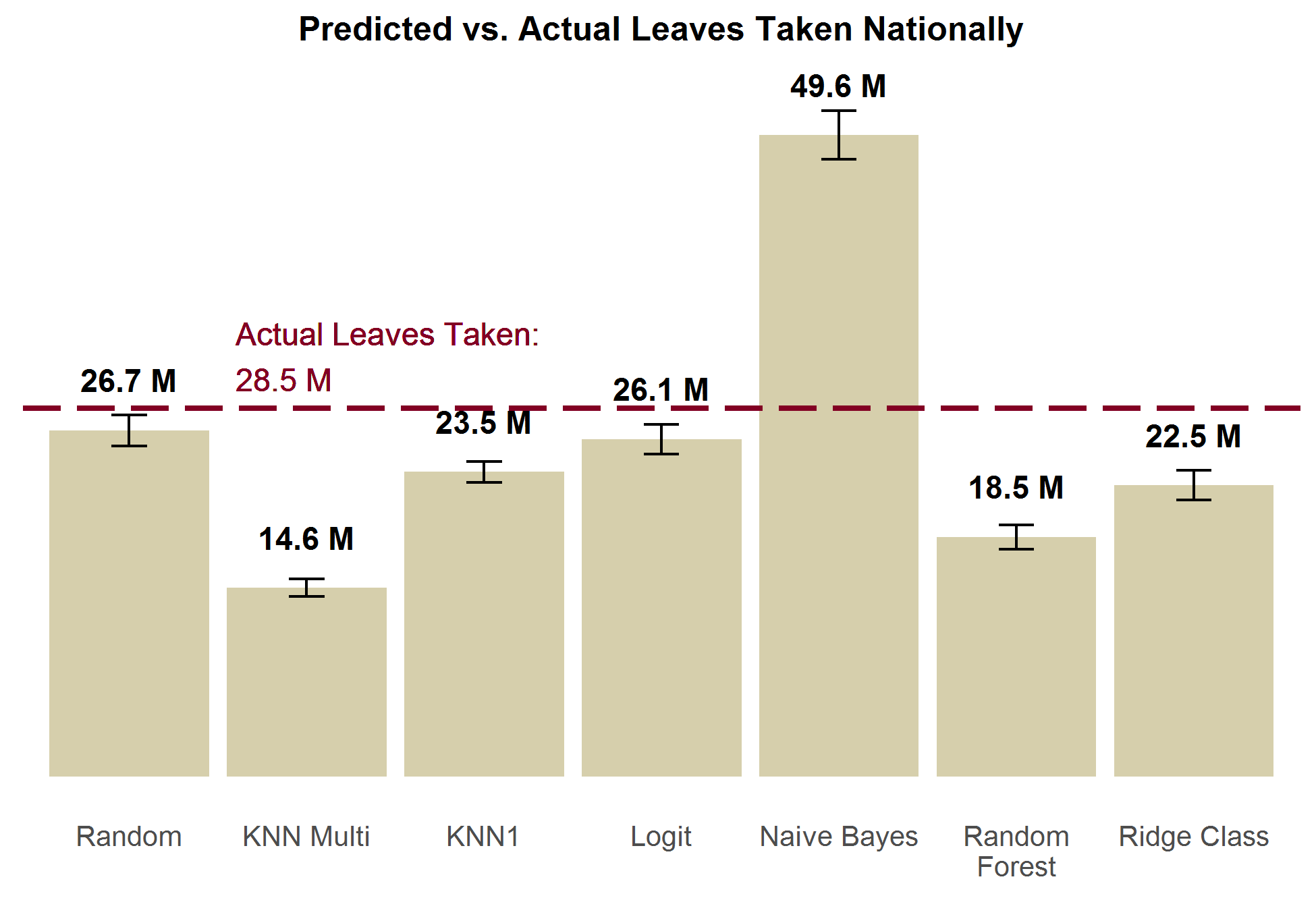
***Leave Takers.*** Exhibit 16 below shows the number of individuals nationally each method predicts to take at least one leave in a 12 month period, and compares that prediction to the actual number of 23.5 million leave takers. As we see, KNN1 hits closest to that mark; no other method captures the actual value within their 95% confidence interval. Logit, KNN multi, random forest, and ridge class all substantially understate the number of leave takers, while Naïve Bayes overstates it (with a prediction of 26.2 million leave takers).

**Exhibit 16.**



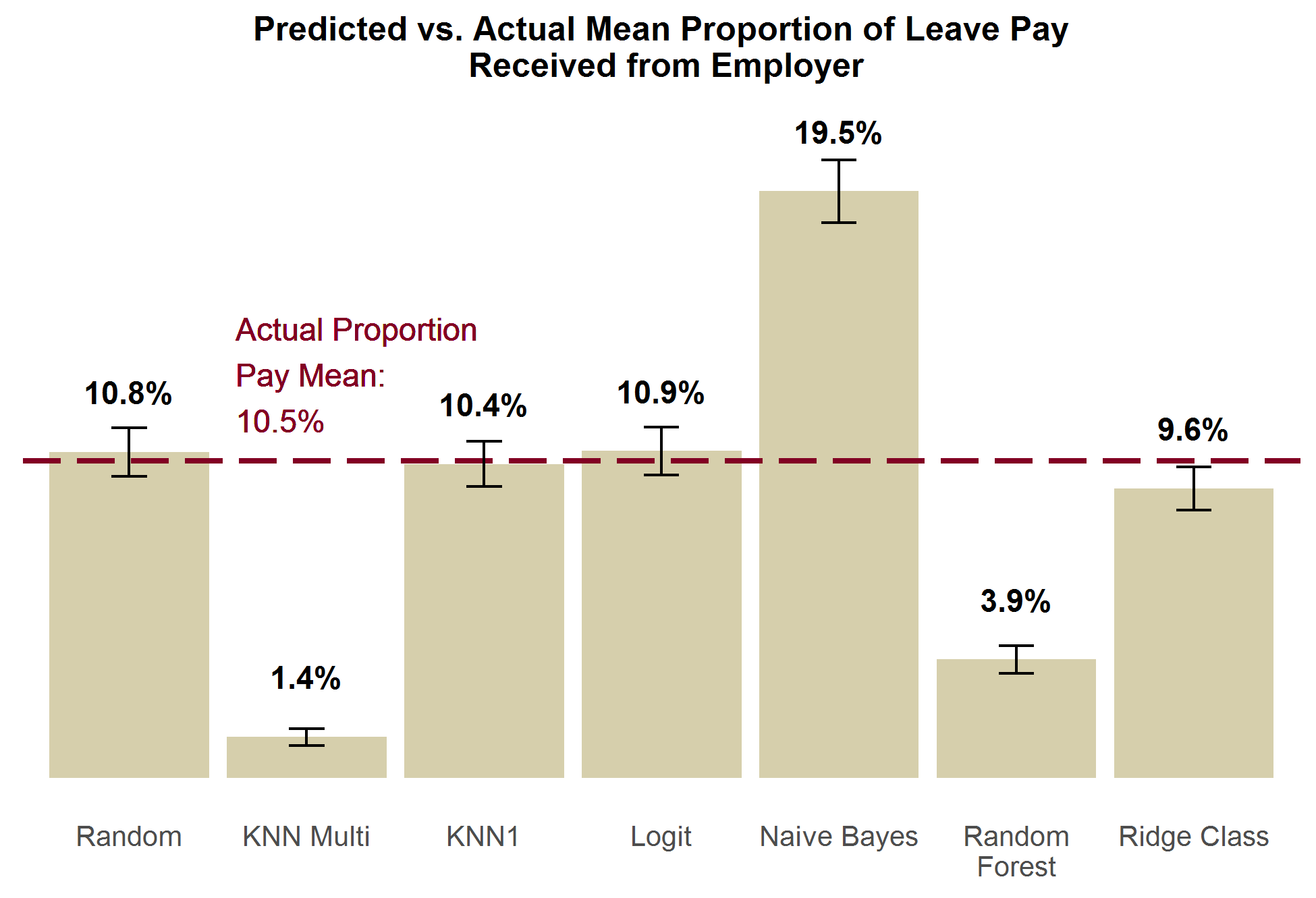
***Leaves Taken.*** Exhibit 17 below is slightly different from Exhibit 16 above. It measures the total predicted *leaves* rather than *leave takers*. There are a number individuals who require multiple leaves, and so these numbers are not the same. Correspondingly, there are more actual leaves taken (28.5 million) than actual leave takers (23.5 million). While no method captures the true value within their estimate’s confidence interval, logit comes closest with a small underestimation of 26.1 million leaves. KNN1 is next with 23.5 million, and ridge class is third with 22.5 million. KNN multi and random forest more drastically understate leaves taken. Naïve Bayes drastically overpredicts number of leaves; predic ting 49.6 million leaves.

**Exhibit 17.**



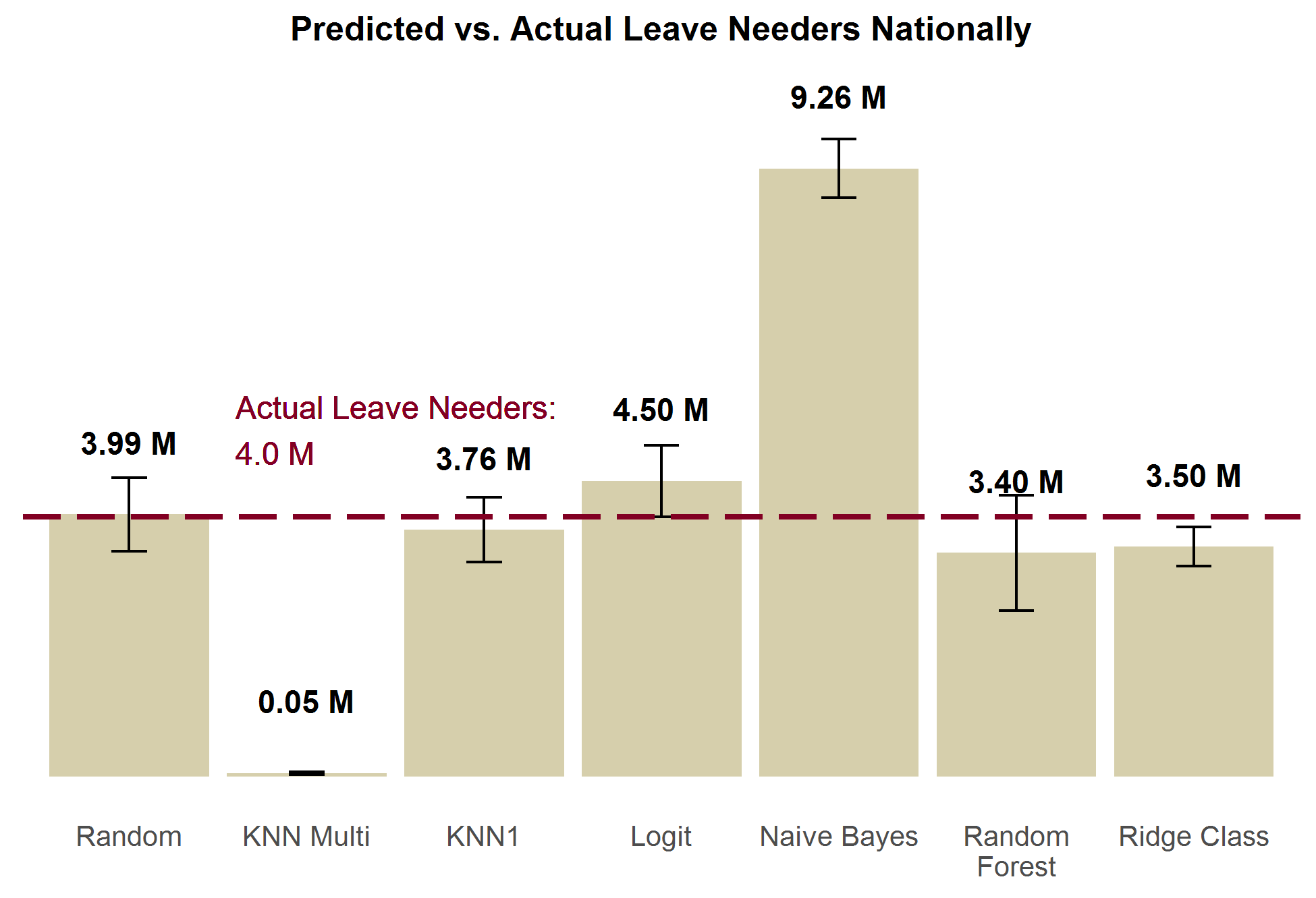
***Proportion of Pay Received from Employer.*** For the proportion of pay received, we compare the average mean predicted for each method in Exhibit 18 below. On average, individuals actually receive about 11% of their wages from their employers. KNN1 and logit methods both have estimates close to the actual value; both capture the actual mean within their estimate’s confidence interval. Ridge class is close with a mean prediction of 9.6%, but undershoots the mean value by a small amount. Random forest and KNN1 both drastically underestimate the proportion of pay received, while Naïve Bayes drastically overstates the proportion of pay received.

**Exhibit 18.**



***Leave Needers.*** Exhibit 10 below displays the predicted versus actual leave needers. There were actually 4 million leave needers in 2011 according to the FMLA survey. KNN1, logit, ridge class, and random forest methods all come close to properly estimating this. KNN multi drastically understates leave needing, and estimates just a handful of individuals will need leave. Naïve Bayes drastically overstates the number of leave needers.

**Exhibit 19.**



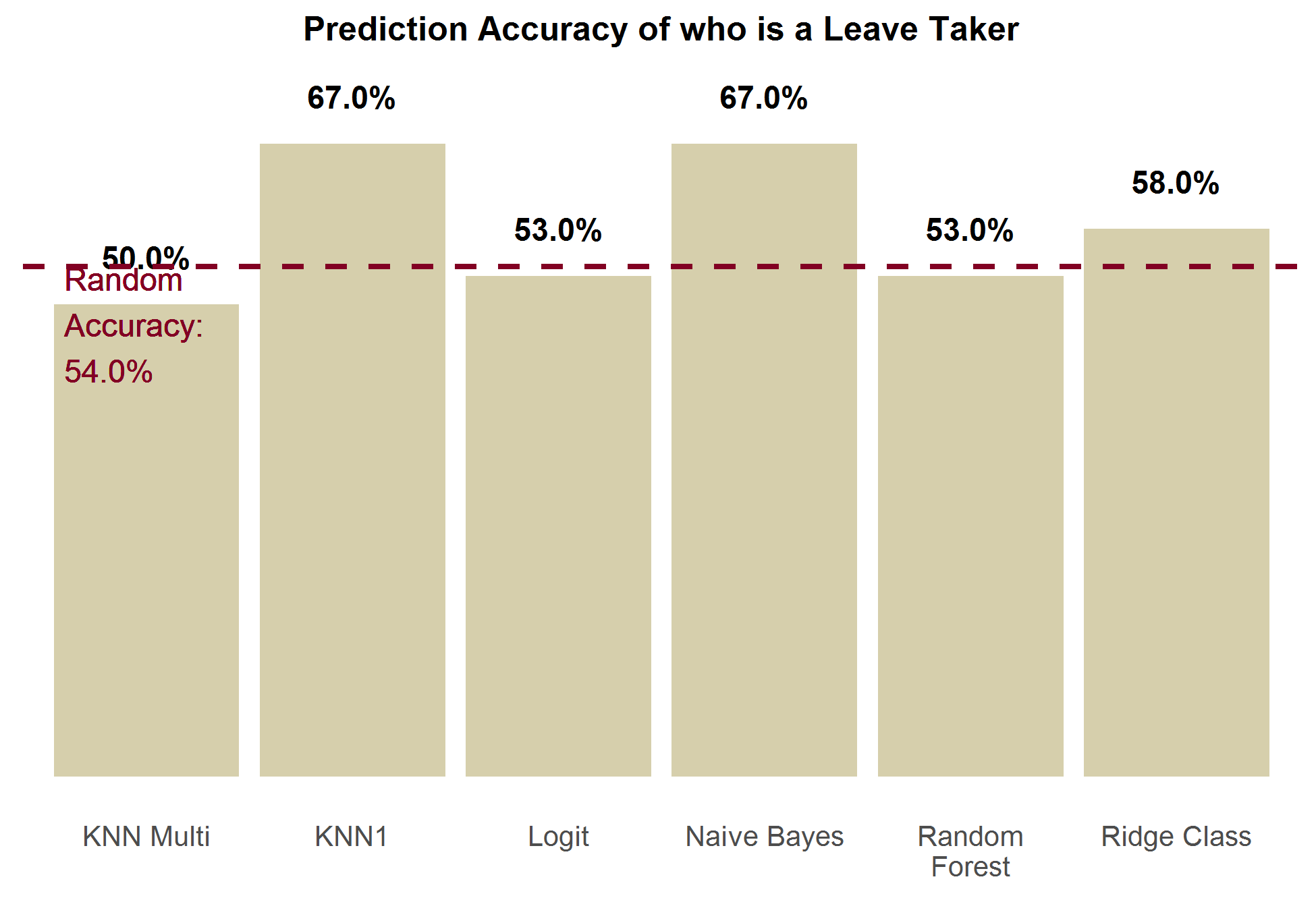
### C.3 FMLA-to-FMLA Individual-Level Performance

The previous subsection looking at aggregate performance of each method; how accurately they predicted “*how many* individuals took leave/needed leave/etc.” But we also want to know how well these models predict “*who* took leave/needed leave/etc.” To find this out, we also tested how accurate the predictions of the methods were at the individual-level, as well as their precision and recall. In this section, we also compare methods’ performance against random draws as baseline performance. The improvement from random draws is illustrative of the marginal gain we have achieved by using the given imputation method.

In this section, the rank order of the methods by different measures is significantly more heterogeneous than the results from the previous two section’s tests. The lack of consistency in these results leaves is a contrast of the conclusive evidence of KNN1’s superiority from the previous two sections. What is most consistent and instructive from these tests is the relatively modest gains in performance models typically exhibit over random draws.

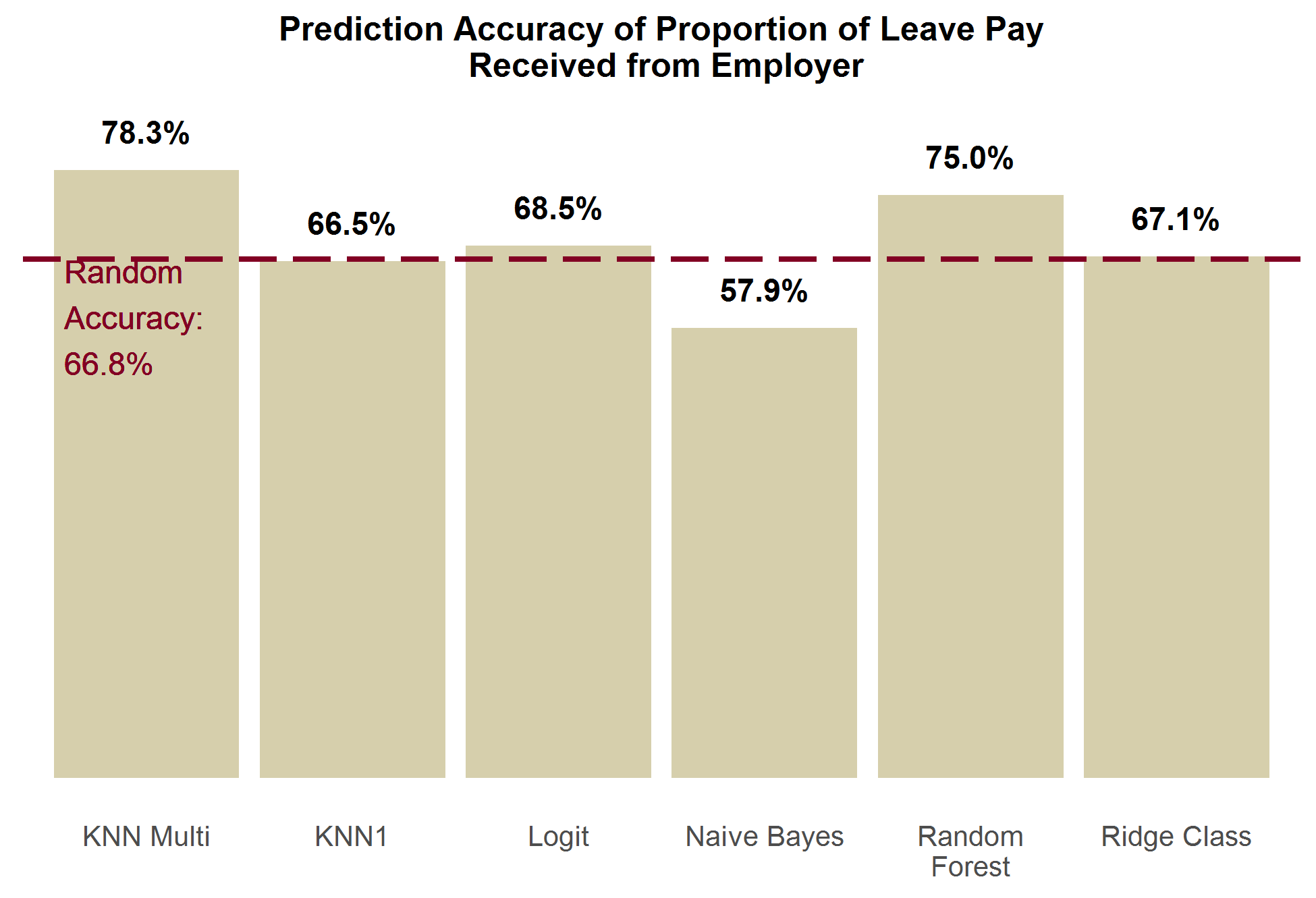
***Leave Takers.*** Exhibit 20 looks at the overall accuracy of each model at predicting leave takers, compared to random draws. Only KNN1 and Naïve Bayes are appreciably better than random draws. Even these two methods still only show modest improvement over random draws, and still often make errors. This is a strong indicator that conditional independence does not hold; there are unobservable characteristics related to leave taking, which leads to biased predictive models. We are only able to use the limited set[[2]](#footnote-2) of overlapping demographic characteristics between the FMLA and ACS surveys; which belies the true complexity of leave taking decisions. These results drive our recommendation to use our model primarily to answer population-level (“how many take leave?”) research questions, and to view answers to individual-level (“who takes leave?”) research questions with caution.

**Exhibit 20.**



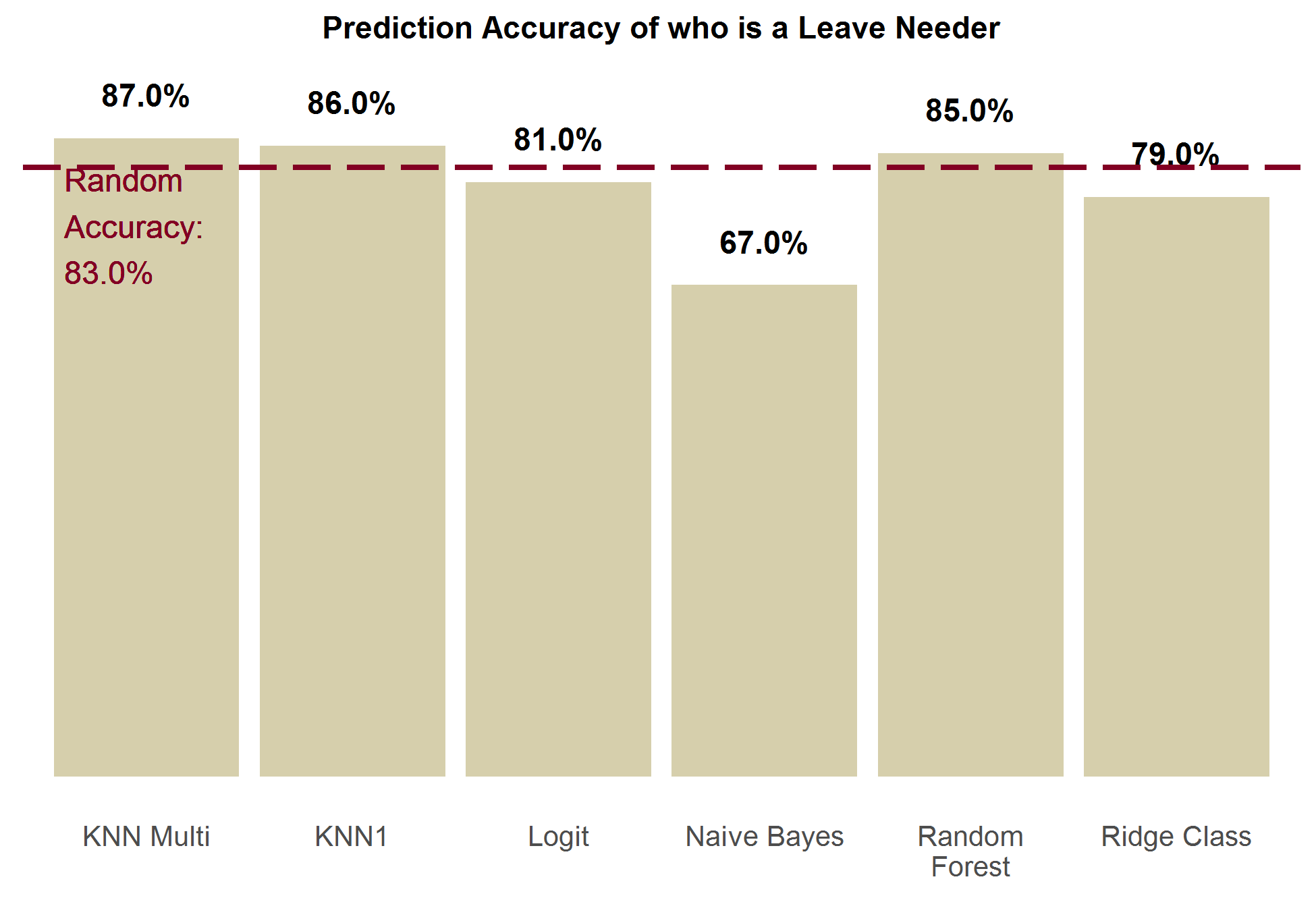
***Proportion of Pay Received from Employer.*** Exhibit 21 shows how each imputation method performs at predicting the correct proportion of leave pay received from their employer, compared to each other and to random draws. This is the proportion of individuals a method predicted the correct proportion of pay value for (out of the 7 possible values). KNN multi is the most accurate, closely followed by random forest. The other 4 methods are all comparable or worse than random draws. Naïve Bayes was the only method to perform worse than random draws.

**Exhibit 21.**

******

***Leave Needers.*** Exhibit 22 shows that no method does much better than random accuracy, and half of the methods actually do worse than random. KNN multi is the best performing method, but only gets an additional 4% better accuracy than random guessing.

**Exhibit 22.**



1. See Saunders, C., A. Gammerman and V. Vovk (1998). Ridge regression learning algorithm in dual variables. [↑](#footnote-ref-1)
2. Each model by default includes all of these overlapping variables. They are: gender, marital status, age, education, race, ethnicity, family income, and presence/absence of children. [↑](#footnote-ref-2)