Machine Learning (IN4320)

Final Report

Volker Strobel (4524187)
August 10, 2016

Abstract

This report presents the techniques and results of a classification problem—predicting whether a person earns more than EUR 40k a year—that involved missing data as well as a mixture of categorical and continuous data. The problem of missing data points was addressed by Multiple Imputation Chained Equations (MICE). Support vector machines (SVM) were used for the classification. The approach was evaluated locally using cross-validation and remotely on a hold-out test-set using the Kaggle platform. The best Kaggle submission resulted in an accuracy of 85.718 %, which corresponds to the 3rd position out of 26 participants in the competition.

1 The Challenge

The Kaggle competition *Final Assignment IN4320*[1] asks participants to predict whether a person earns more than EUR 40k a year from $D = 13$ predictor variables. The provided dataset was created by a census bureau. The task can be divided into the following three challenges:

- Imputation of missing data points
- Managing a mixture of categorical and continuous independent variables
- Classification of the output variable

In this report, the terms *wealthy* or *label 1* describe a sample with income greater than EUR 40k and *non-wealthy* or *label 0* are used otherwise.

In Section 2, I analyze and visualize the structure of the data to set the stage for the classification pipeline. In Section 3, the used methods for data imputation and classification are detailed. Section 4 presents and discusses the results obtained during cross-validation and on Kaggle.

[1] `https://inclass.kaggle.com/c/final-assignment-in43202`

2 Analysis & Ideas

To motivate the used classifier, design, and technique choices, in this section, the provided dataset is analyzed and possible concepts and ideas are introduced.

In total, 13 features are given, of which 8 are categorical and 5 are continuous (Table 1). The training set consists of 10500 samples and the test set has 38342 samples, resulting in a ratio of approx. 1 : 3.7. The test set is thus considerably larger than the training set. Since transductive learning is permitted, the information in the test set—information about the distribution of the variables—could help to increase the classification performance.

Categorical	Continuous
work class	age
education	number of years of education
marital status	income from investment sources
occupation	losses from investment sources
relationship	working hours per week
race	
sex	
native country	

Table 1: Overview of the used features

10500 samples constitute the training set of which 2500 (23 %) were labeled as wealthy and 8000 (77 %) as non-wealthy. If we assume that the training and test set were randomly sampled from the entire dataset, therefore, the predictions on the test set should reflect this ratio. Additionally always predicting non-wealthy should give a 0–1 loss of 77 % under this assumption, which can be used as a baseline for the classifier performance. The assumption could be tested by probing the test set with a '0-only' submission. This ratio might be useful for setting the class-weights of a classifier or for determining the threshold in majority voting (see Section 3.4).

In total, approx. 20 % of the data values are missing, with approx. 23 % for the variables *workclass* and *occupation*, 20 % for *country of origin*, and 19 % for the remaining variables. The percentage and pattern of missing data between the training and the test set showed no apparent difference. Only 6 % of samples in the training set and in the test set are complete cases (i.e., have no missing values), making adequate treatment of missing data a crucial step. I assume that the higher values for the variables *workclass*, *occupation*, and *country* stem from missing values in the original dataset and the missing values for the remaining variables were just introduced.

Given the description of the dataset and the total number of samples (48842), the dataset may be the UCI Adult Data set [9]. This would underline the pattern of missing data (higher values for workclass, occupation, and country of origin). A possible method would be to use the labeled samples of the UCI dataset and match them with the given test set, which should give an accuracy of 1.0. However, this method is not as straight-forward as it may seem. The UCI dataset is split in a different manner into training and test set (amount of training samples: 32561, test samples: 16281). Additionally, the categorical variables in the UCI dataset are encoded as strings, while in the given dataset, they are encoded as integers. Therefore, one would have to find a mapping from strings to integers. The large amount of missing data might impede this try. Since using the UCI Adult dataset might defeat the goal of this assignment, I did not take any steps in this direction. A detailed explanation of the data set[2] shows that the best performing classifier is a Forward Sequential Selection (FSS) naive Bayes model with an accuracy of 85.95 %. The classifiers were trained after removing unknown values (7 % of values had missing values; training set size: 30162, test set size: 15060). This accuracy can be used as an indicator of a good performance on the given dataset. However, there are two differences to the given dataset: (i) the original UCI dataset had a lower amount of missing data, and (ii) the UCI dataset had a higher number of training samples.

3 Methods

3.1 Overview of the Classification Pipeline

In Figure 1, the complete classification pipeline is visualized. In the following subsections, each step will be explained in more detail. The process starts with combining the training and the test set to yield a large dataset with missing values. Using Multivariate Imputation by Chained Equations (MICE), the missing values are imputed and five training and test sets are obtained. Five classifiers are trained using the different training sets; for each training set all test sets are used for obtaining the 'preliminary hypotheses'. By means of majority voting with a modifiable threshold θ, the single hypotheses are pooled to yield the final prediction vector.

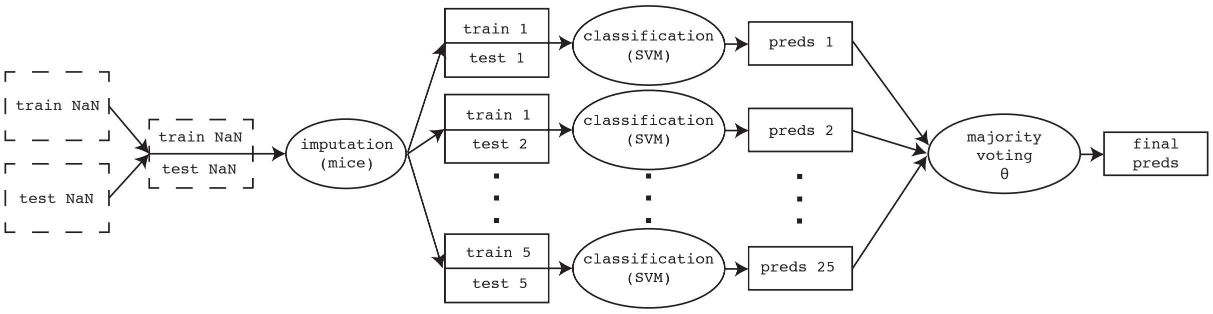

Figure 1: The figure illustrates the classification pipeline. Matrices are displayed as boxes and methods as ellipses; dashed lines indicate missing values.

3.2 Dummy Variables

Seven of the twelve measured variables are qualitative (workclass, education, marital status, occupation, relationship, sex, native country)—they are measured only at the nominal level. Since measurements on the nominal level do not allow for a particular ordering, a proxy method has to be used to make them suitable for some statistical models. To this end, I defined dummy variables that take the value 0 or 1 to indicate if a certain category is present. Defining n dummy variables for the n different possible values of a categorical feature allows for capturing the full information in the original unmodified dataset and for using qualitative data in a straight-forward manner. Transforming the dataset greatly increased its size from $D = 13$ to $\tilde{D} = 104$ predictor variables.

[2]`http://www.cs.toronto.edu/~delve/data/adult/adultDetail.html`

3.3 Missing Data Values

Missing data values are a common problem in statistics due to the failure of sensors, or the concealment of data. Thus, methods have been put forth for the imputation of missing data points, such as case deletion or single mean imputation [14]. However, such simple methods often discard useful information and are likely to give a rather poor result if a large part of the data is missing. The general goal is to find unbiased estimates of the missing values. The bias depends on the type of "missingness":

- Missing Completely At Random (MCAR)
  In the case of MCAR missingness, there is no dependence structure between the missing data and any other values. Listwise deletion leads to unbiased analyzes.
- Missing At Random (MAR)
  In the case of MAR, the reason for missingness depends on the value of other non-missing predictor variables. Unbiased estimates can be obtained using the multiple imputation method [7], which creates multiple versions of filled-in datasets.
- Missing Not At Random (MNAR)
  In the case of MNAR, the reason for missingness depends on non-available information. Such a case might for example occur, if people with high income refuse to reveal their income. Imputation of MNAR requires explicit modeling of the reason for missingness to yield unbiased estimates.

For obtaining unbiased estimates with listwise deletion, the data must be MCAR. Little's MCAR test [10] is a statistical test with the null hypothesis that the data are MCAR. I used it at the significance level $\alpha = 0.05$ to analyze the interaction structure of the variables. The statistic was significant for the combined dataset consisting of the training and the test set ($\chi^2 = 28018.84$, $p < 0.001$, $df = 15639$). Therefore, there is evidence that the missing values in the dataset are not MCAR; the null hypothesis is rejected. It is not possible to distinguish between MAR and MNAR data using observed information only [15]. Consequently, I assumed that the data are MAR, since imputation of MNAR data would need further information about the reason for missingness, which was not given. Therefore, data points were imputed using the multiple imputation method by a technique called Multivariate Imputation by Chained Equations (MICE) [3] using the `R` package `mice` (Version 2.25). By using multiple imputations, the algorithm incorporates the statistical uncertainty in the imputed values. MICE is a flexible approach that can also handle continuous and categorical variables. The general idea is to statistically model the conditional probability of each missing variable given the remaining variables. In detail, the MICE algorithm works as follows [3, 1]:

1. In the beginning, a simple imputation is carried out. To this end, all missing data points are replaced by random sampling with replacement from the observed datapoints.
2. One variable is selected at random, for example *occupation*. An intermediate model is built, using the remaining variables as predictors and *occupation* as target value. The chosen model is dependent on the target value: I used logistic regressions for categorical data and linear regressions for numerical data. The missing values in the variable *occupation* are replaced by the predictions of the model.
3. The previous step is executed for all variables with missing data. For each variable, the

model is trained using both the already imputed values and the existing values. Once all variables have been predicted, one cycle is complete.

4. Several cycles are performed to stabilize the imputation results. I used $c = 5$ cycles.
5. The entire procedure is executed multiple times to yield several imputed datasets. Due to the random factors, the imputed values will be different, while the non-missing data entries will be the same in all datasets. I used $m = 5$ imputations.

The choices $c = 5$ and $m = 5$ were made due to practical limitations, considering time limitations and avoiding overflow errors; for the imputation $\tilde{D} \cdot m \cdot c = 104 \cdot 5 \cdot 5 = 2600$ intermediate models were built. A value of m between 5 and 10 is a common value for the used method [12] and a value of $c = 5$ is considered "adequate" [16]. However, recent research shows that a higher value of m might be beneficial [17], which could be tested, if a more powerful machine would be at hand.

Importantly, the MICE algorithm was used on the total dataset, consisting of the training and the test set. This should increase the quality of the imputed values since more training examples can be used for building the intermediate models (Step 2 in the algorithm outline).

In the following, the two statistical models—linear regression and logistic regression—used for the imputations are briefly described:

Linear regression: For the linear regression (continuous variables), a design matrix $\mathbf{X}$ is built from the predictor variables x_{ij} (jth feature of the ith sample):

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & x_{23} & \dots & x_{2d} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & x_{n3} & \dots & x_{nd} \end{bmatrix} \tag{1}$$

The first column in the matrix enables fitting the intercept in the regression. The solution for the weights $\hat{\beta}$—which minimizes the squared-error loss function—is obtained using the normal equation $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}\mathbf{y}$, where $\mathbf{y}$ is the vector consisting of the target values. Predictions for one variable y_i are made with $\hat{y}_i = X_i\hat{\beta}$, where X_i describes the features of the ith sample.

Logistic regression: Logistic regression (binary variables) uses a logit link function to link the linear predictor $X_i\beta$ to the probability of *label 1*:

$$\Pr(y_i = 1|X_i) = \text{logit}^{-1}(X_i\beta) = \frac{1}{1 + e^{-X_i\beta}} \tag{2}$$

Logistic regression can be trained via stochastic gradient descent (SGD). For this, the following update function is used—with a sufficiently small learning rate α and $p = \text{logit}^{-1}(X_i\beta_t)$:

$$\beta_{t+1} := \beta_t + \alpha(y - p)X_i \tag{3}$$

The difference $y - p$ is the error made at time step t by predicting y from X_i using the weight vector β_t. For predicting the target value (0 or 1) of the sample X_i, the hypothesis function $h(X_i)$ is used, which transforms the class probabilities into target values:

$$h(X_i) = \begin{cases} 1 & \text{if } \Pr(y_i = 1|X_i) > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

3.4 Pooling

I trained models and made predictions on all possible combinations of imputed training and test sets. This results in 25 "preliminary hypotheses" $\tilde{h}_{ij}$ for the ith sample: $\tilde{\mathbf{h}}_\mathbf{i} = (\tilde{h}_{i,j})_{j\in\{1,2,\ldots,25\}} = [\tilde{h}_{i,1}, \tilde{h}_{i,2}, \ldots, \tilde{h}_{i,25}]$. These predictions had to be aggregated to obtain one final submission. For the aggregation, I used the following majority voting function on the preliminary hypotheses:

$$\mathbf{h}_\mathbf{i} = \text{vote}_\theta(\tilde{\mathbf{h}}_\mathbf{i}) = \begin{cases} 1 & \text{if } \sum_{j=1}^{25} h_{ij} \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The function predicts the class *wealthy*, if at least $\theta \in \{0, 1, 2, \ldots, 26\}$ preliminary hypotheses are 1. Introducing θ in the function was motivated by the local cross-validation (next section): the default predictions of the classifier were "conservative"—meaning that it predominantly predicted the class *non-wealthy*—and only 54.0 % of actual *wealthy* samples were correctly classified. The threshold θ serves as modulator for the "conservatism" of the classifier: the higher θ is, the less likely it is that the final prediction *wealthy* will be made. The threshold $\theta = 13$ would represent standard majority function with a 50 % majority. Figure 2 shows the trade-off between specificity and sensitivity for increasing values of θ. It can be seen that the accuracy is not greatly affected by the threshold θ: a higher sensitivity is directly traded-off with a smaller specificity. The best accuracy was achieved at $\theta = 9$.

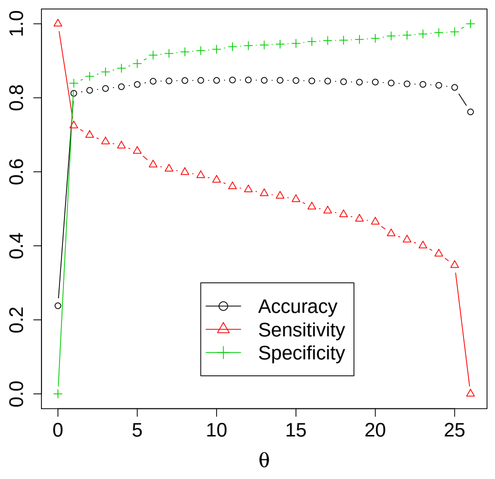

Figure 2: Accuracy, sensitivity, and specificity in dependence of the threshold θ. The accuracy stays roughly the same for $6 \leq \theta \leq 25$. In contrast, the sensitivity falls and the specificity rises for increasing values of θ.

3.5 Cross-Validation

To evaluate submissions and determine the ranks of participants, the Kaggle challenge used the 0–1 loss. While the score on the public leaderboard is the best validation of the methods employed, only one submission could be submitted per participant and day. To evaluate my approach more frequently, I implemented a local ($k = 5$)-fold cross-validation. In each fold, a model was trained on $\frac{4}{5}$ of the data and tested on the remaining samples from all five imputed datasets. Therefore, $k \cdot m \cdot m = 5 \cdot 5 \cdot 5 = 125$ accuracy values were obtained. Per fold, the predictions for the $m \cdot m = 25$ "preliminary hypotheses" were aggregated using majority voting with the threshold parameter θ. To obtain the final cross-validation accuracy per model, the mean accuracy of the five folds was calculated. The cross-validation achieved similar results to the public Kaggle evaluation but underestimated the public performance in all cases (Table 2).

Method	train imps.	test imps. per model	θ^*	Local Score (%)	Public Score (%)
AdaBoost	1	1	/	83.886	84.205
Random Forest	5	1	3	84.038	85.056
SVM	5	1	3	84.413	85.494
AdaBoost	5	5	13	84.933	85.434
SVM	5	5	9	85.305	85.718

Table 2: Comparing local cross-validation and public Kaggle scores. The column *training imputations* specifies the number of imputed datasets for building the models; the column *test imputations per model* specifies how many test sets were used to get the "preliminary hypotheses". θ^* indicates the value of the threshold θ that resulted in the highest accuracy in the cross-validation.

Each fold in the cross-validation contained 2100 test samples and 8400 training samples. Since the classes *wealthy* and *non-wealthy* are not equally distributed, a confusion matrix can give a more detailed picture of the classifier performance. After predicting the labels for all five folds, the following confusion matrix was obtained:

		Predicted class		
		wealthy	non-wealthy	Total
Actual class	wealthy	$TP = 1351$	$FN = 1149$	$TP + FN = 2500$
	non-wealthy	$FP = 546$	$TN = 7454$	$FP + TN = 8000$
	Total	$TP + FP = 1897$	$FN + TN = 8603$	$N = 10500$

From the confusion matrix, sensitivity and specificity can be calculated:

$$\text{Sensitivity} = \frac{TP}{TP + FN} = 54.0\,\% \tag{6}$$

$$\text{Specificity} = \frac{TN}{TN + FP} = 93.2\,\% \tag{7}$$

$$\tag{8}$$

These statistics show that almost all non-wealthy samples are correctly classified, while only slightly more than half of the wealthy samples are correctly classified. The confusion matrix implies that the classifier is "conservative": it avoids classifying samples as *wealthy.* The high overall accuracy is mainly caused by correctly classifying non-wealthy samples.

3.6 The Classifier

Section 3.5 shows the different classifiers that were compared. The best performing classifier was a support vector machine (SVM) [5]. For running the algorithm `Python 2.7.11` was used with the package `scikit-learn` in Version 0.17.0.

The choice of an SVM was motivated by the following points:

1. SVMs work “out-of-the-box” on many different problems [13].
2. SVMS work in high-dimensional spaces ($\tilde{D} = 104$)
3. SVMs aim to minimize the generalization error by finding a decision boundary that maximizes the margin.
4. SVMs allow for *transductive learning* using transductive SVMs. Therefore, the information in the unlabeled test data can be incorporated in the training process, possibly improving the accuracy.
5. SVMs showed the best performance in the local cross-validation.

If D is the number of features, SVMs build a $(D-1)$-dimensional hyperplane that tries to maximize the margin between the two classes. The support vectors are the samples on the margin for the positive class and the negative class. The maximum-margin hyperplane is the hyperplane “right in the middle” of the support vectors.

The classifier is of the form $f(\mathbf{x}) = w^T\mathbf{x} + b$ and the corresponding classification function is $h(\mathbf{x}) = sign(w^T\mathbf{x}+b)$, with $\mathbf{x}$ being the feature vector, w the coefficient vector and b the intercept term. The target variables y_i for SVMs are -1 (*non-wealthy*) and 1 (*wealthy*), instead of 0 and 1. To train the classifier, the goal is to find w, b such that the resulting decision boundary maximizes the margin. Therefore, the following optimization problem—known as the *primal representation*— is solved:

$$\min ||\mathbf{w}||^2 + C\sum_{i=1}^{N}\zeta_i \tag{9}$$

subject to the constraints:

$$\mathbf{w}^T\mathbf{x}_i \geq +1 - \zeta_i, \text{for } y_i = +1$$

$$\mathbf{w}^T\mathbf{x}_i < -1 + \zeta_i, \text{for } y_i = -1$$

$$\zeta_i \geq 0, \forall i$$

To allow for non-linearly separable datasets and make the SVM less sensitive to outliers, the variables ζ_i introduce “slackness” into the constraints, such that not every sample has to lie at the correct side of the margin. If $0 < \zeta_i \leq 1$, the sample is between the margin and the correct side of hyperplane; if $\zeta > 1$, the sample is misclassified. The hyperparameter C trades off structural error and training error: a value close to 0 specifies that violating the margin constraint—that each sample has to be at the correct side of the margin—is of not much consequence; greater values put more emphasis on satisfying the margin constraint for each data point.

To find w and b, it is often more convenient to use the *dual representation*, which can be derived from the primal form [13]:

$$\arg\max_{\alpha} \sum_j \alpha_j - \frac{1}{2}\sum_{j,k}\alpha_j\alpha_k y_j y_k(\mathbf{x_j}\cdot\mathbf{x_k}) \tag{10}$$

subject to the constraints:

$$\alpha_j \geq 0$$

$$\sum_j \alpha_j y_j = 0$$

The variable $y_i \in \{-1, 1\}$ represents the class of x_i and α is the weights vector that assigns a weight α_j to each data point. The *support vectors* are the samples for which the weights a_j are non-zero. This optimization problem can be solved using quadratic programming, for which several libraries exist. I used the solution provided by the `libsvm` library [4].

To extend SVMs to non-linear classification, the *kernel trick* can be used, with which data can be embedded in a higher-dimensional space. The kernel $K(\mathbf{x_j}, \mathbf{x_k})$ is then applied to the dot products of feature vectors in Equation 10. Therefore, the equation becomes:

$$\arg\max_{\alpha} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k K(\mathbf{x_j}, \mathbf{x_k}) \tag{11}$$

Practically, this means that the problem of finding a linear separator is shifted to a higher dimensional feature space. Mapping back this linear separator to the original feature space results in a non-linear decision boundary. For the given problem, a Gaussian radial basis function kernel was used:

$$K(\mathbf{x_j}, \mathbf{x_k}) = \exp(-\gamma ||\mathbf{x_j} - \mathbf{x_k}||^2) \tag{12}$$

The two hyperparameters of the SVM, C and γ, were chosen using grid-search in the cross-validation.

Before training, each variable in the combined dataset, consisting of training and test set, was scaled to zero mean and unit variance using the following formula:

$$x' = \frac{x - \bar{x}}{\sigma} \tag{13}$$

In the formula, x is the original value, x' the new value, and $\bar{x}$ and σ mean and standard deviation of the variable. Scaling ensures that features with a greater range do not dominate features with smaller ranges [8].

3.7 Transductive Learning

Since the test data is known at training time, this information can be used for using semi-supervised methods than can better capture the underlying distribution. I made use of the self-learning method: at first, the unlabeled data was labeled by predicting their labels using a SVM trained on the provided training set. Afterward, another SVM was trained using the combined dataset consisting of training and the test set and the final predictions for the test set were made. The approach did not improve the classification performance.

Two more sophisticated approaches—using the implementations of `semisup-learn`[3]— Contrastive Pessimistic Likelihood Estimation (CPLE) [11] and Transductive SVMs [2, 6] were terminated prematurely since training was not finished after 24 hours.

4 Results & Discussion

In total, I made 10 submissions during the competition. The best submission resulted in an accuracy of 85.718 %, which corresponds to the 3rd position out of 26 participants when finishing this report.

[3] `https://github.com/tmadl/semisup-learn`

In summary, the main challenge lay in imputing the missing data values. Afterward, standard classification techniques could be used. The threshold parameter θ of the majority voting had no substantial influence on the classification performance. Also, different classifiers did not greatly affect the achieved accuracy: support vector machines, random forest classification, and AdaBoost classifiers yielded similar results, as shown in Table 2.

The use of self-supervised learning did not have a positive effect on the classification performance; this is in line with other studies, showing that semi-supervised methods do not necessarily lead to improved performances and can even lead to a lower accuracy [18]. It might be that semi-supervised methods would have had a greater effect, if the number of training samples would have been substantially smaller.

Since the ground truth labels of the test set were not revealed, error statistics could only be computed based on the 0–1 loss on the public leaderboard and on the local cross-validation. The restriction of one submission per day increased the difficulty of the validation. My public score of 85.718 %—which is 0.00876 below the best performing score in the competition (86.594 %) and 0.00232 below the best performing method of the Delve project [4]—is an indicator that little improvement will be possible beyond my approach. Compared to the UCI dataset, the given dataset had a smaller amount of labeled data and more missing values, which made the Kaggle classification task more challenging. Using standard state-of-the-art classifiers, the maximum possible performance with the provided dataset will be possibly capped below 100 % due to mistakes during data acquisition, such as deliberate misinformation or transcription errors. Moreover, the used variables might only have limited explanatory power: additional variables, such as political view, morale, or number of children may be needed to increase the explainable variation.

Due to the concealment of categorical variables, I did not consider manual feature engineering such as grouping certain levels of the categorical (dummy) variables.

Many steps in the classification pipeline were time-consuming, taking several hours to complete. In the future, more processing power could help to increase the classification performance. For example, using a larger amount of imputed datasets could capture the uncertainty in the imputed values to a greater degree and another method for transductive learning could help to increase the classification performance.

My code for this competition can be found at:
`https://github.com/Pold87/ml-final-ass`

Word Count: 3412

[4]`http://www.cs.toronto.edu/~delve/data/adult/adultDetail.html`

References

[1] Melissa J. Azur et al. “Multiple imputation by chained equations: What is it and how does it work?” In: *International Journal of Methods in Psychiatric Research* 20.1 (2011), pp. 40–49.

[2] Kristin Bennett, Ayhan Demiriz, et al. “Semi-supervised support vector machines”. In: *Advances in Neural Information processing systems* (1999), pp. 368–374.

[3] Stef Buuren and Karin Groothuis-Oudshoorn. “mice: Multivariate imputation by chained equations in R”. In: *Journal of Statistical Software* 45.3 (2011).

[4] Chih-Chung Chang and Chih-Jen Lin. “libsvm: A library for support vector machines”. In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 27:1–27:27.

[5] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.

[6] Fabian Gieseke et al. “Fast and simple gradient-based optimization for semi-supervised support vector machines”. In: *Neurocomputing* 123 (2014), pp. 23–32.

[7] Moniek C.M. de Goeij et al. “Multiple imputation: dealing with missing data”. In: *Nephrology Dialysis Transplantation* 28.10 (2013), pp. 2415–2420.

[8] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. “A practical guide to support vector classification”. In: (2003).

[9] M. Lichman. *UCI Machine Learning Repository.* 2013. URL: `http://archive.ics.uci.edu/ml`.

[10] Roderick JA Little. “A test of missing completely at random for multivariate data with missing values”. In: *Journal of the American Statistical Association* 83.404 (1988), pp. 1198–1202.

[11] Marco Loog. “Contrastive pessimistic likelihood estimation for semi-supervised classification”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.3 (2016), pp. 462–475.

[12] Patrick Royston, Ian R. White, et al. “Multiple imputation by chained equations (MICE): implementation in Stata”. In: *Journal of Statistical Software* 45.4 (2011), pp. 1–20.

[13] Stuart Jonathan Russell et al. *Artificial intelligence: a modern approach.* Vol. 2. Prentice Hall, 2003.

[14] Joseph L. Schafer and John W. Graham. “Missing data: Our view of the state of the art.” In: *Psychological Methods* 7.2 (2002), p. 147.

[15] Jonathan A.C. Sterne et al. “Multiple imputation for missing data in epidemiological and clinical research: Potential and Pitfalls”. In: *British Medical Journal* 338 (2009).

[16] Stef Van Buuren, Hendriek C. Boshuizen, Dick L. Knook, et al. “Multiple imputation of missing blood pressure covariates in survival analysis”. In: *Statistics in Medicine* 18.6 (1999), pp. 681–694.

[17] Ian R. White, Patrick Royston, and Angela M. Wood. “Multiple imputation using chained equations: Issues and guidance for practice”. In: *Statistics in Medicine* 30.4 (2011), pp. 377–399.

[18] Xiaojin Zhu. *Semi-supervised learning literature survey.* 2005.