MODELING REPORT FOR CUSTOMER INCOME LEVEL ANALYSIS FOR DIFFERENT MARITAL STATUS

ZHEYUAN WU, DIJKSTRA LIU

1. Introduction

In the era of data-driven decision-making, analyzing customer behavior has become paramount for businesses aiming to fine-tune their marketing strategies and enhance customer satisfaction. This study delves into a rich dataset encompassing various attributes related to customers' demographic profiles such as, *Year_Birth*, *Education*, *Marital_Status*. We aim the exploration of how marital status (*Marital_Status*) correlates with annual household income, which in turn influences spending patterns across various product categories such as wines, fruits, and other goods over a biennial span. This inquiry not only aims at mapping out correlations but also endeavors to model predictive insights that could guide stakeholders in crafting more personalized, data-informed approaches towards engaging diverse customer segments.

A model that can classify the income level of each user from marketing dataset could be useful for various stakeholders, including:

(1) Marketing and advertising companies:
   (a) These companies could use such a model to segment their target audience more effectively based on predicted income levels.
   (b) They could tailor their marketing campaigns, messaging, and advertising content to resonate better with different income groups, increasing the likelihood of successful conversions and sales.
(2) E-commerce and retail businesses:
   (a) By predicting customers' income levels, these businesses could personalize product recommendations, promotions, and user experiences accordingly.
   (b) They could offer more relevant products, services, or content to different customer segments, improving customer satisfaction and potentially increasing sales.

We will also use Ensemble Learning and Model Averaging in analyzing the data. By training diverse models, such as k-nearest neighbors, artificial neural networks, decision trees, random forests, naive Bayes classifiers, and support vector machines (SVMs), we can combine their predictions through ensemble learning techniques like bagging, boosting, or stacking. Ensemble methods often outperform individual models by leveraging the strengths of each model and compensating for their weaknesses, leading to improved accuracy and robustness.

2. Results

2.1. **Model performance on standardized data.** The metric we choose is Accuracy, which is commonly used as an evaluation metric for prediction problems, including predicting customer income levels based on consumer personality analysis datasets for different marital statuses. However, the suitability of using accuracy as the primary metric depends on the specific characteristics of the problem and the dataset. Here are some reasons why accuracy might be a reasonable choice for this particular problem:

Balanced class distribution: If the classes (income levels) in the dataset are relatively balanced across different marital statuses, accuracy can provide a good overall measure of the model's performance. When classes are imbalanced, accuracy may not be the best metric as it can be skewed towards the majority class.

Equal misclassification costs: Accuracy assumes that the cost of misclassifying an instance into any other class is equal. In the context of customer income level prediction, this assumption might be reasonable if the consequences of misclassifying a customer into a higher or lower income level are similar.

Interpretability: Accuracy is a straightforward and interpretable metric. It represents the proportion of correctly classified instances, which can be easily understood by stakeholders and decision-makers.

Baseline comparison: Accuracy provides a baseline for comparing the performance of different models or algorithms. If the goal is to achieve the highest possible accuracy, it serves as a useful metric for model selection and optimization.

K-fold cross-validation provides a more accurate and reliable estimate of a model's performance on unseen data compared to using a single train-test split. By training and evaluating the model on multiple folds, the performance estimate is less prone to bias or variance due to the specific way the data was partitioned.

To get a reliable result for binary classification, we use 5-fold validation for tuning the hyperparameters of our models. The hyperparameters are recorded under each experimentation. Some results with large variance are not recorded and requires more investigation.

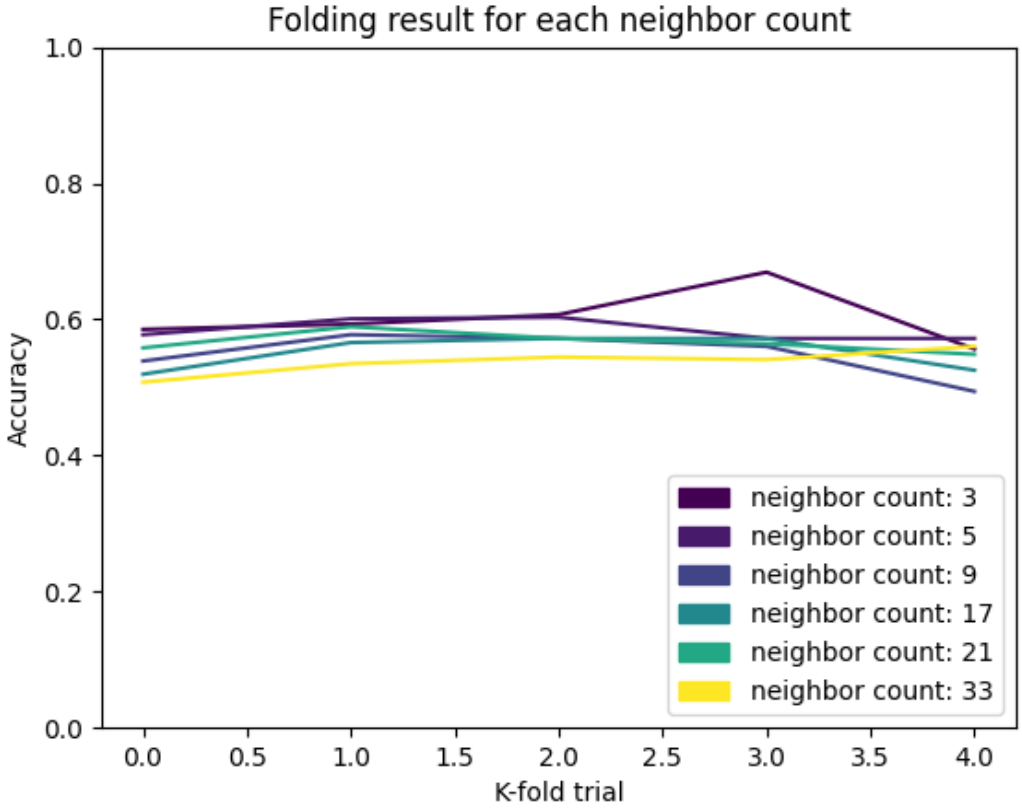

FIGURE 1. 5-fold visualization for different hyperparameters in KNN

The result for 5-fold and prediction accuracy are as follows:

Model performance for income level from sigle population (non-reduced)			
Model Name	Training time (seconds)	Training precision	Testing precision
SVM	0.0110	0.92	0.91
ANN	2.94	0.93	0.92
KNN	0.0000	0.90	0.89
RFC	0.1406	0.93	0.91

In this population, the (potentially) best hyperparameters for income level from sigle population (non-reduced) are as follows:

SVM best: $C = 1, kernel = linear$ For the SVM model: C = 1: The parameter C is a regularization parameter that helps control the trade-off between achieving a low error on the training data and minimizing the model complexity for better generalization. A C value of 1 provides a good balance, preventing the model from fitting the training data too closely (overfitting). Kernel = Linear: The linear kernel is simple and effective for linearly separable data. Using a linear kernel often requires less computation and avoids the risk of overfitting when the data is not complex.

ANN best: $unit = 5, activation = relu, lr = 0.01, batch\_size = 16$.

For the ANN model: Units = 5: This refers to the number of neurons in the hidden layer. Having 5 units is a choice that can capture sufficient complexity in the data without overcomplicating the model, which can help in avoiding overfitting and reducing computational demand. Activation = ReLU (Rectified Linear Unit): ReLU is a popular activation function because it introduces non-linearity into the model without affecting the gradients too much, which helps in avoiding the vanishing gradient problem common with other activation functions like sigmoid or tanh. Learning Rate = 0.01: This is a moderate learning rate that helps ensure the model learns at a reasonable pace; not too fast to skip optimal solutions, and not too slow to impede the learning process. Batch Size = 16: This batch size is small enough to allow the model to update its weights frequently, which can lead to faster convergence. Small batches can also provide a regularizing effect and lower the risk of overfitting.

Model performance for income level from no long time relationships population (non-reduced)			
Model Name	Training time (seconds)	Training precision	Testing precision
SVM	0.0020	0.84	0.92
ANN	0.24	0.94	0.91
KNN	0.0000	0.88	0.86
RFC	0.4865	0.90	0.8

The (potentially) best hyperparameters for income level from no long time relationships population (non-reduced) are:

SVM: Best parameters: $\{'C' : 100, 'kernel' : 'rbf'\}$

ANN: Best hyperparameters: $\{'units' : 5, 'activation' : 'relu', 'learning_rate' : 0.1, 'epochs' : 20, 'batch\_size' : 16, 'accuracy' : 0.9125909752547308, 'time' : 2.878108024597168\}$

KNN: Best parameters: $\{'n_neighbors' : 33\}$

RFC: Best parameters: $\{'max_samples' : 0.7, 'n_estimators' : 256\}$

Model performance for income level from married population (non-reduced)			
Model Name	Training time (seconds)	Training precision	Testing precision
SVM	0.0020	0.87	0.92
ANN	0.95	0.92	0.92
KNN	0.0000	0.90	0.91
RFC	0.3014	0.91	0.92

The (potentially) best hyperparameters for income level from married population (non-reduced) are:

SVM: Best parameters: $\{'C' : 100,' kernel' :' rbf'\}$

ANN: Best hyperparameters: $\{'units' : 5,' activation' :' relu',' learning\_rate' : 0.1,' epochs' : 20,' batch\_size' : 16,' accuracy' : 0.9125909752547308,' time' : 2.878108024597168\}$

KNN: Best parameters: $\{'n\_neighbors' : 33\}$

RFC: Best parameters: $\{'max\_samples' : 0.7,' n\_estimators' : 256\}$

2.2. **Dimension reducing techniques.** After reducing the variables, most models gets worse performance. The model performance are as follows:

Model performance for income level from single population (reduced)			
Model Name	Training time (seconds)	Training precision	Testing precision
SVM	0.0110	0.91	0.89
ANN	2.9526	0.893	0.90
KNN	0.0000	0.86	0.83
RFC	0.0427	0.89	0.91

The (potentially) best hyperparameters for income level from single population (reduced) are:

ANN: Best hyperparameters: $\{'units' : 10,' activation' :' tanh',' learning\_rate' : 0.01,' epochs' : 20,' batch\_size' : 16\}$

SVM: Best hyperparameters: $\{'C' : 10,' kernel' :' poly'\}$

KNN: Best parameters: $\{'n\_neighbors' : 3\}$

RFC: Best parameters: $\{'max\_samples' : 0.9,' n\_estimators' : 16\}$

Model performance for income level from no long time relationships population (reduced)			
Model Name	Training time (seconds)	Training precision	Testing precision
SVM	0.0020	0.83	0.89
ANN	0.6231	0.86	0.82
KNN	0.0000	0.84	0.83
RFC	0.469	0.88	0.86

The (potentially) best hyperparameters for income level from married population (reduced) are:

KNN: Best parameters: $\{'n\_neighbors' : 33\}$

RFC: Best parameters: $\{'max\_samples' : 0.5,' n\_estimators' : 16\}$

Model performance for income level from married population (reduced)			
Model Name	Training time (seconds)	Training precision	Testing precision
SVM	0.007	0.89	0.9
ANN	0.9544	0.892	0.86
KNN	0.0000	0.87	0.9
RFC	1.0349	0.91	0.92

The (potentially) best hyperparameters for income level from married population (reduced) are:

KNN: Best parameters: $\{'n\_neighbors' : 33\}$

RFC: Best parameters: $\{'max\_samples' : 0.9,' n\_estimators' : 512\}$

As the analysis show, it is practical to predict a person's income level with those given data. The companies could use such a model to segment their target audience more effectively based on predicted income levels. They could tailor their marketing campaigns, messaging, and advertising content to resonate better with different income groups, increasing the likelihood of successful conversions and sales.

In E-commerce and retail businesses. With sufficient data, we can predict customers' income levels, these businesses could personalize product recommendations, promotions, and user experiences accordingly.

They could offer more relevant products, services, or content to different customer segments, improving customer satisfaction and potentially increasing sales.

3. Methods

3.1. **Data Prepossessing.**

First we split up the income level by median in the sample data set (around 50000)

Then use the second option to split the population to three by maternal status:

In long-time relationship

$$Married, Together$$

Not in long-time relationship

$$YOLO, Alone, Single$$

Lost partner from long-time relationship

$$Divorced, Widow, Absurd$$

We want to try to classify these population's income level based on other data provided.

The data structure of processed data are as follow:

Categorical data are:

- $Education$: Customer's education level
- $Marital\_Status$: Customer's marital status

Numerical data are:

- $Year\_Birth$: Customer's birth year
- $Income$: Customer's yearly household income
- $Kidhome$: Number of children in customer's household
- $Teenhome$: Number of teenagers in customer's household
- $Recency$: Number of days since customer's last purchase
- $MntWines$: Amount spent on wine in last 2 years
- $MntFruits$: Amount spent on fruits in last 2 years
- $MntMeatProducts$: Amount spent on meat in last 2 years
- $MntFishProducts$: Amount spent on fish in last 2 years
- $MntSweetProducts$: Amount spent on sweets in last 2 years
- $MntGoldProds$: Amount spent on gold in last 2 years
- $NumDealsPurchases$: Number of purchases made with a discount
- $NumWebPurchases$: Number of purchases made through the company's website
- $NumCatalogPurchases$: Number of purchases made using a catalogue
- $NumStorePurchases$: Number of purchases made directly in stores
- $NumWebVisitsMonth$: Number of visits to company's website in the last month
- $Complain$: 1 if the customer complained in the last 2 years, 0 otherwise

Among those data, normally distributed are:$Year\_Birth$, $Recency$

3.2. **Model Selection.** Wen each of the model, we tune the following parameters:

(1) k-nearest neighbors is a model that predicts class of a given value based on the data points around it.

k-nearest neighbors is better than naive-bayes classifier because - it can be used to fit non linear relationships. - it is easy to implement and interpret.

However k-nearest neighbors has following weakness comparing to naive-bayes classifier because - it need to run with the data set, which required more storage than naive-bayes classifier - it generally has longer training time.

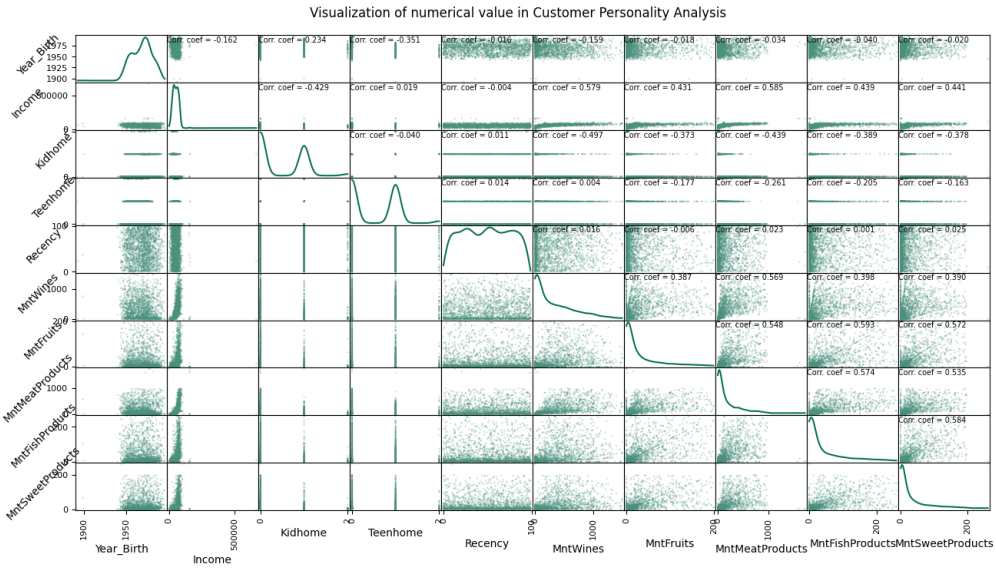

FIGURE 2. An ugly figure to show multidimensional distribution of the dataset

(2) Random Forest is a model that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Random forest is better than decision-tree because - it uses ensemble learning by averaging to improve the predictive accuracy and control over-fitting. - each of the tree is easy to explain.

Nevertheless, Random forest has following weakness comparing to decision-tree because - generally has longer training time. - generally requires more storage required comparing to decision

(3) Artificial Neural Networks (ANN) are computational models inspired by the human brain, consisting of layers of interconnected nodes that process input data through activation functions to make predictions or classifications. They are capable of learning complex patterns from data through a process called training, where the model adjusts its internal parameters to minimize error.

Artificial Neural Networks (ANN) are often preferred over simpler models like logistic regression because:

they can model highly complex and non-linear relationships within the data. they excel in tasks with large and diverse datasets, adapting through deep learning to achieve better performance.

Artificial Neural Networks (ANN) have the following weaknesses compared to simpler models like logistic regression:

They require significant computational resources for training, which can be time-consuming and costly, especially for large networks. They are prone to overfitting, particularly in cases where the amount of data is limited relative to the complexity of the model.

(4) Support Vector Machines (SVM) are supervised learning models that analyze data for classification and regression tasks. They work by finding the hyperplane that best divides a dataset into classes with the largest margin, thus providing a robust prediction model especially effective in high-dimensional spaces.

Support Vector Machines (SVM) are favored over k-nearest neighbors in some cases because:

they are effective in high-dimensional spaces, where the distance between data points is crucial for classification. they offer a clear margin of separation, which helps to reduce the risk of overfitting in contrast to more local or memory-intensive methods.

Support Vector Machines (SVM) also exhibit certain disadvantages compared to models like logistic regression:

They are not very efficient with large datasets as the training time can scale poorly, making them less suitable for larger-scale applications. They can be sensitive to the choice of kernel and regularization parameters, which can make tuning the model complex and require careful selection to achieve optimal performance.

3.3. **Hyperparameter Tuning.**

The process of hyperparameter tuning is central to optimizing the performance of machine learning models. Hyperparameters are the adjustable parameters that control the model training process, as opposed to the internal parameters determined by the training data itself. Proper selection and adjustment of these parameters can significantly enhance model performance, especially in complex datasets.

(1) For the hyper-parameter tuning, KNN:
We will use different neighbor counts for KNN. We'll test values
$n\_neighbors = 3, 5, 9, 17, 21, 33$

(2) Random Forest:
We will use the number of trees. We'll test values
$n\_estimators = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512$
We will test $max\_samples$ passed to each tree, the number of samples to draw from X to train each base estimator. We'll try the fractions
$max\_samples = 0.1, 0.3, 0.5, 0.7, 0.9$

(3) For the Artificial Neural Network (ANN), you can explore the following hyperparameter settings during your experimentation:
Units: Number of neurons in each layer. We tested different layer sizes with values units=$[5, 10]$. Activation: Type of activation function used in the neurons. I'll try both $activation = [relu, tanh]$.
Learning Rate: The step size at each iteration while moving toward a minimum of a loss function. Possible values to test are $learning\_rate = [0.1, 0.01]$.
Epochs: Number of times the learning algorithm will work through the entire training dataset. For this test, I'll use epochs=[20].
Batch Size: Number of samples propagated through the network before the update of internal model parameters. I'll experiment with $batch\_size = [16]$.

(4) For the Support Vector Machine (SVM), the hyperparameter grid you could explore includes:
C (Regularization parameter): The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty. I'll test values $C = [0.1, 1, 10, 100]$. Kernel: Specifies the kernel type to be used in the SVM algorithm. Options to test include $kernel = [linear, poly, rbf, sigmoid]$.