

**Machine Learning**

Homework 1

Supervisor:

Dr. Hashemi

Ali Mahmoodi

Atefe Rajabi

Winter 2024

Index:

[Problem Definition](#ProblemDefinition)

[Implementation](#Implementation)

[Questions & Answers](#QA)

[Result](#Result)

**Problem Definition**

In the domain of machine learning, concept learning can be understood as the task of inferring a boolean-valued function from training examples of its input and output. Often represented as a form of classification, concept learning involves identifying the underlying rule or concept that categorizes the input data into positive and negative examples. The primary challenge in concept learning is to construct a model that can accurately predict the classification of unseen instances based on the knowledge derived from a set of examples provided during the training phase.

Concept learning sits at the core of many machine learning applications, where the goal is to acquire a thorough understanding of the underlying patterns within the data. The algorithms that approach this task can vary widely in complexity and methodology, ranging from simple memorization to sophisticated generalization techniques. This document presents an implementation and examination of four distinct concept learning methods: Rote Learning, Find-S Algorithm, Candidate Elimination, and List Then Eliminate.

**Implementation:**

**`RoteLearning` Class:**

- This class implements rote learning, a form of learning where the model memorizes the entire training dataset.

- The `fit` method stores the training data and their corresponding labels.

- The `predict` method compares each instance in the validation set (`val\_data`) with all instances in the training set and returns the label of the matching instance. If no exact match is found, it returns `None`.

**`FindS` Class:**

- Implements the Find-S algorithm, which finds the maximally specific hypothesis.

- The `fit` method initializes the most specific hypothesis (`self.h`) based on positive instances in the training data. It does so by iterating over each attribute; if all positive instances share the same value for an attribute, that value is used in the hypothesis; otherwise, a '?' is used to denote any value is acceptable.

- The `predict` method checks if the data matches the hypothesis. If the data matches or is generalized by the hypothesis (`'?'`), the method returns '+', otherwise '-'.

**`CandidElimination` Class:**

- Implements the Candidate Elimination algorithm.

- Requires an instance of the `FindS` class during initialization to make use of the specific hypothesis from `FindS` as a starting point.

- The `get\_all\_h` method generates all possible hypotheses by mixing specific values and '?' for attributes.

- The `evaluate` method is used to check if the instances match the hypotheses.

- The `fit` method initializes the specific hypothesis like `FindS` and then generates general hypotheses. It keeps those general hypotheses that do not cover any negative instance from the training data. The variable `self.vs` represents the version space that is consistent with the training examples.

- The `predict` method checks which hypotheses in the version space are consistent with the validation data. It aggregates predictions from all hypotheses and chooses the most common prediction; if there's a tie, it returns `None`.

**`ListThenEliminate` Class:**

- This class is a variant of the Candidate Elimination algorithm where it starts with a hypothesis space that includes all possible hypotheses.

- The `get\_all\_h` method generates this comprehensive list of hypotheses.

- The `evaluate` method, like in `CandidElimination`, checks for consistency between data and hypotheses.

- The `fit` method iterates over all possible hypotheses and retains only those that are consistent with all the training data.

- The `predict` method functions similarly to `CandidElimination`, where it aggregates predictions from the remaining hypotheses in the version space to predict on validation data.

**Questions & Answers:**

1. **What is a hypothesis?**

A hypothesis is a specific set of rules or conditions that attempts to capture the underlying pattern in the data. A hypothesis represents a possible relationship between inputs (often called features or attributes) and the desired output. When a new input is given to the hypothesis, it makes a prediction or classification based on its learned rules. A hypothesis is the output of the training process that can estimate a certain target function in this case the target function is to determine whether a sample is positive or negative.

1. **What does it mean when we say a hypothesis is consistent with training samples?**

A hypothesis is ***consistent*** with the training examples if it correctly classifies these examples. A Hypothesis h is consistent with a set of training examples D if and only if h(x) = c(x) for each example <x,c(x)> in D.

Consistent(h,D) (

1. **Are our training samples consistent?**

Yes, since all the training samples are categorized in a correct label category by the Founded hypotheses. In most practical learning problems there is some chance that the training examples will contain at least some errors or noise. Such inconsistent sets of training examples can severely mislead FIND-S, given the fact that it ignores negative examples. We would prefer an algorithm that could at least detect when the training data is inconsistent and, preferably, accommodate such errors.

If a hypothesis can be found for the datum, the sample is consistent.

**Randomly select another 8 samples from our dataset, are those consistent as well?**

No, for candidate Elimination

Random\_samples:

['private', 'high school grad', 'single', 'white', 'Male'],

['private', 'high school grad', 'married', 'black', 'Male'],

['self-employed', 'high school grad', 'married', 'white', 'Female'],

['self-employed', 'college grad', 'married', 'white', 'Female'],

['private', 'college grad', 'married', 'white', 'Female'],

['self-employed', 'high school grad', 'single', 'white', 'Male'],

['private', 'college grad', 'married', 'black', 'Male'],

['private', 'college grad', 'married', 'other', 'Female']

Random\_labels:

['+', '+', '+', '-', '+', '-', '-', '+']

Specific bound:

['?', '?', '?', '?', '?']

General bound: None

Status: 'inconsistent'

1. **What’s a hypothesis space?**

The complete set of all possible hypotheses that can be formulated for a given problem within the constraints of a particular representation.

The hypothesis space can be very large or even infinite, depending on how many features are considered and the complexity of the models allowed.

Unique values for each feature:

['government' 'private' 'self-employed']

['college grad' 'high school grad']

['married' 'single']

['black' 'other' 'white']

['Female' 'Male']

instance space: 3\*2\*2\*3\*2=72

hypothesis space: 2^72

conjuctive hypothesis space: 4\*3\*3\*4\*3=16\*27=432

1. **What’s a version space?**

The version space, denoted V SH,D, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with the training examples in D.

V SH,D

1. **What is a query sample in candidate elimination algorithm?**

A query sample represents a specifically chosen instance by the learner that is then presented to an external oracle (such as nature or a teacher) to obtain its correct classification (positive or negative). This process is a part of active learning, where the learner actively selects the samples from which it learns, rather than passively receiving pre-labeled examples.

The purpose of choosing query samples is to effectively and efficiently narrow down the version space—the set of all hypotheses that are consistent with the observed examples—by eliminating hypotheses that do not correspond to the oracle's classification of the queried instances. A good query sample is one that maximally reduces the version space, ideally dividing it as evenly as possible, because this strategy leads to the most efficient learning progress.

Where the version space is defined by the hypothesis space for the Enjoysport concept, the mentioned instance (Sunny, Warm, Normal, Light, Warm, Same) is identified as a good query. This is because it has the property that it is classified as positive by some hypotheses in the version space and negative by others. Depending on the oracle's classification of this instance, the learner can significantly reduce the version space—either by generalizing the S (specific) boundary if the instance is positive, or by specializing the G (general) boundary if the instance is negative. This process effectively narrows down the possible hypotheses, facilitating a more efficient approach towards identifying the target concept.

In general, the optimal query strategy in concept learning is to select instances that divide the current version space into two equal parts, based on the oracle's possible classifications. This is analogous to employing a binary search strategy, where each query optimally reduces the search space by half, allowing the learner to converge on the correct hypothesis with the minimum number of queries. When it's not possible to precisely halve the version space, the strategy aims to balance the division as closely as possible, acknowledging that more queries may be needed than in the ideal binary reduction scenario.

**What is a good query sample for the version space we found?**

In our case, we have provided an example query sample: ['private', 'college grad', 'married', 'white', 'Male']. To determine if this is a good query sample, we look at the current state of the version space.

Our version space consists of:

- The most specific hypothesis (S): ['?', 'college grad', 'single', '?', '?']

- The most general hypothesis (G): [['?', 'college grad', '?', '?', '?']]

For the query sample to be effective, it should be consistent with some hypotheses in the version space but inconsistent with others. This way, when we receive the classification for the sample, we can update the version space by eliminating inconsistent hypotheses and refining others.

1. Comparison with the most specific hypothesis (S):

- S suggests that being a 'college grad' is necessary, and the sample matches this criterion.

- S suggests that 'single' is a necessary criterion, but the sample says 'married'. This inconsistency means that if the oracle classifies this sample as positive (i.e., it fits the concept), we will need to generalize the S hypothesis to no longer exclude married individuals.

2. Comparison with the most general hypothesis (G):

- G is quite open, with '?' indicating any value is acceptable for all attributes except for 'college grad', which must be a match. Since the sample includes 'college grad', it is consistent with G.

If the oracle classifies this query sample as a positive instance, we would:

- Update S to be more general since the current S does not cover the query sample (specifically, we’d need to change 'single' to '?').

- No change to G as it already covers this sample.

If the oracle classifies it as a negative instance, we would: (in this case)

- Update G to exclude this negative instance. Since G currently matches this sample on 'college grad', we would need to introduce constraints in G that exclude this specific negative sample while remaining as general as possible.

1. **Can all the above methods predict class labels for the test samples? Which ones can’t? Why?**

1. Rote Learning: Rote learning is a memorization technique based on repetition. In the context of machine learning, rote learning would imply remembering exact examples from the training data without generalization. Such a method can predict class labels for test samples only if the test instances exactly match the instances from the training set. It cannot generalize to unseen instances because it does not derive any generalized rules from the training data.

2. Candidate Elimination: This algorithm maintains a version space that consists of all the hypotheses that are consistent with the training examples. It incrementally updates the boundaries of this space as it receives new examples. Since the algorithm maintains a set of general hypotheses (G) and a set of specific hypotheses (S), it can make predictions for unseen instances by checking if an instance falls within the version space. However, the algorithm may not provide a definitive prediction if the version space still contains significantly different hypotheses. If the version space is sufficiently converged (i.e., S and G are close to each other), it can predict class labels with more confidence.

3. Find-S: Find-S algorithm maintains only the most specific hypothesis that is consistent with the positive examples seen so far. It starts with the most specific hypothesis possible and generalizes it step by step with each positive instance it encounters. Find-S cannot predict labels for test samples that are negative examples because it is designed to focus only on positive examples and does not account for negative ones in its generalization process. This means that Find-S will always predict the positive class for any test sample that matches the final hypothesis, and it has no basis for prediction otherwise, as it ignores information about negative examples during training.

1. **How accurate are the class predictions using each of the above methods?**

For validation Samples:

['private', 'college grad', 'single', 'black', 'Male']

['private', 'college grad', 'married', 'white', 'Male']

Rote Learning:

predicted Labels: [None, None]

accuracy for Rote Learning: 0.0

Find-S:

predicted Labels: ['+' '-']

accuracy for Find-S: 1.0

Candidate Elimination

predicted Labels: ['+' None]

accuracy for Candid ELimination: 0.5

List than Elimination:

predicted Labels: ['+' None]

accuracy for List than Elimination: 0.5

**Result:**

number of samples: 32561

number of features: 15

Unique values for label:

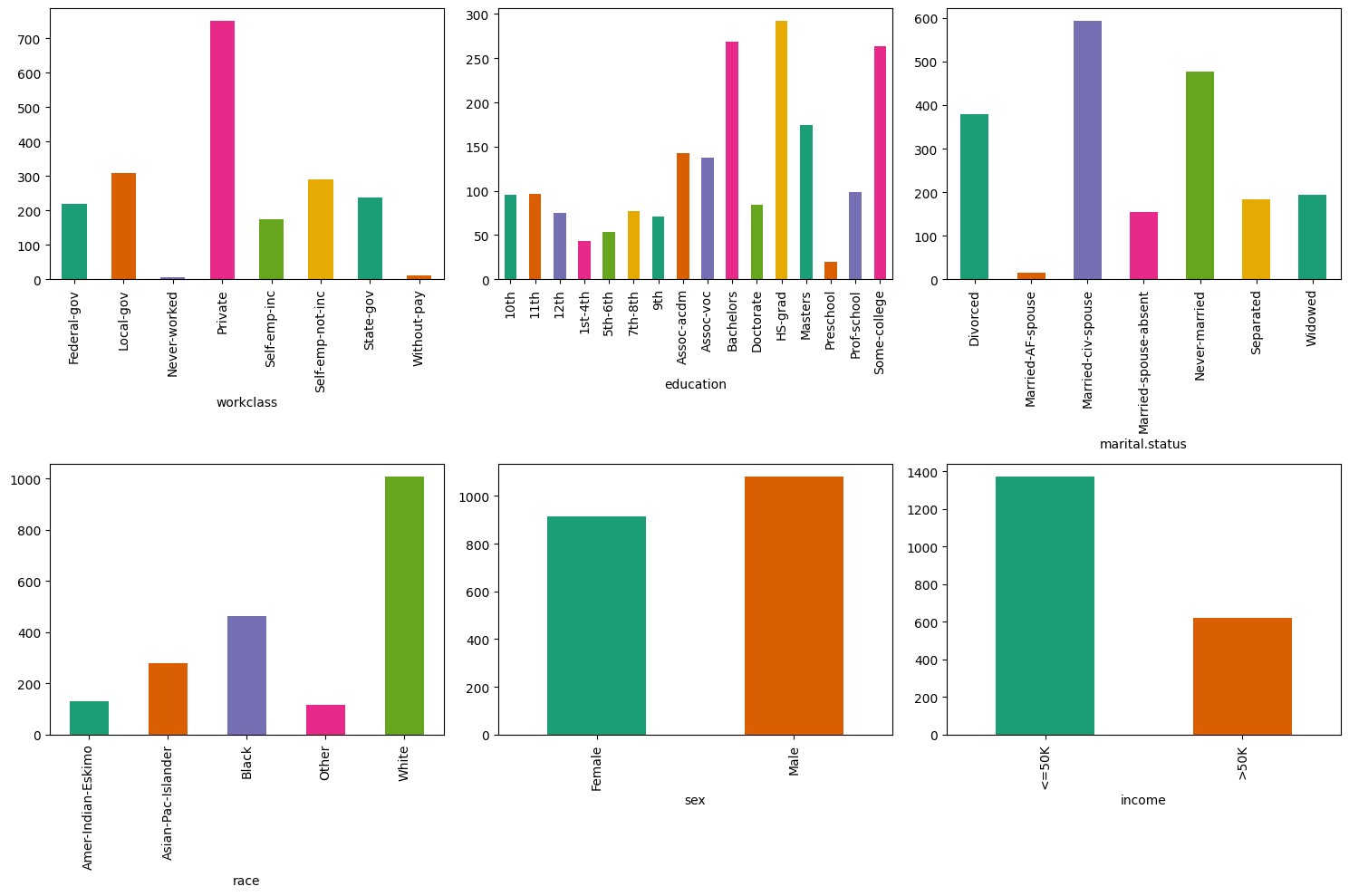
['<=50K', '>50K']

Number of duplicates before dropping: 32561

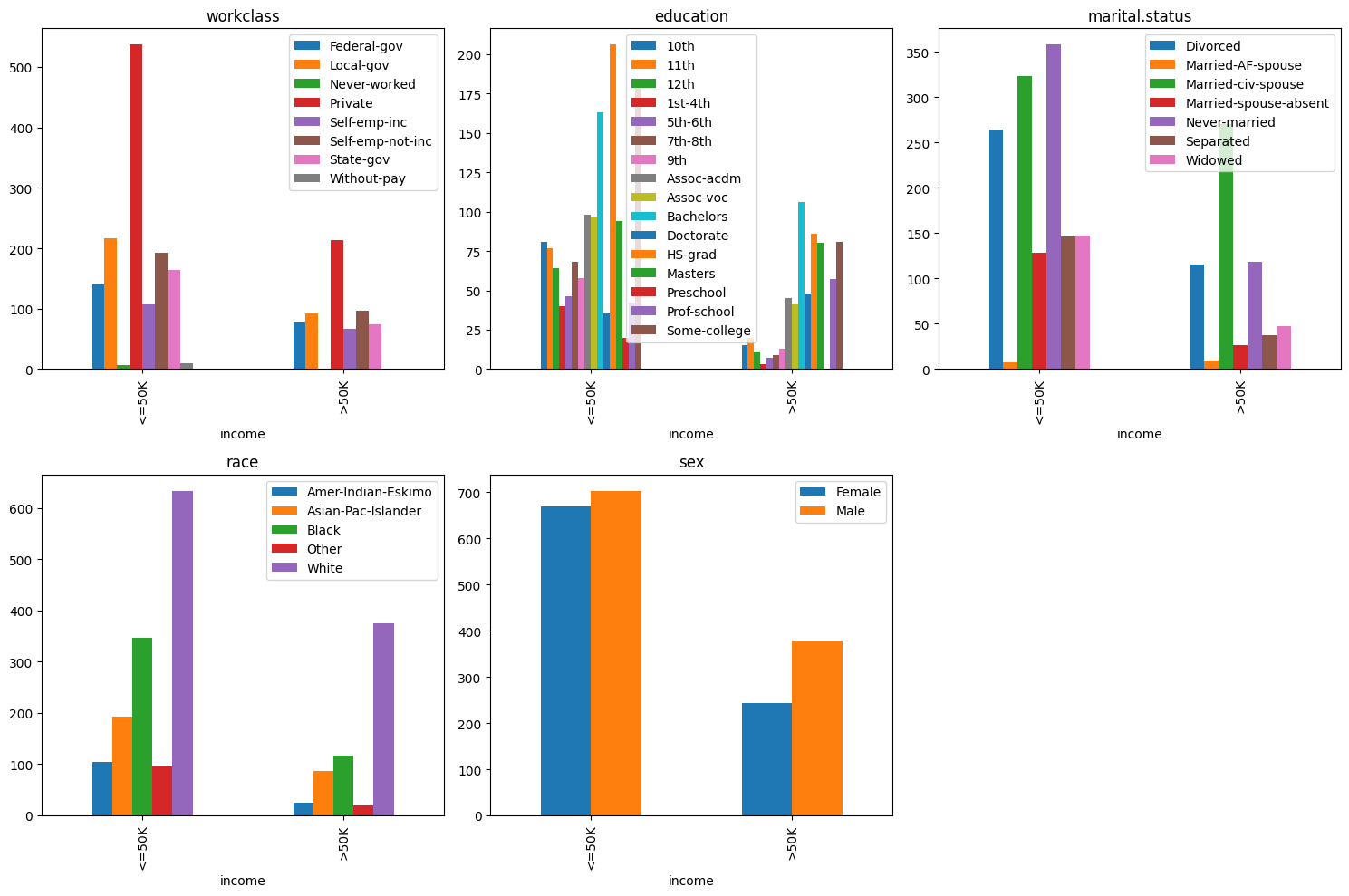
Number of duplicates after dropping: 2300

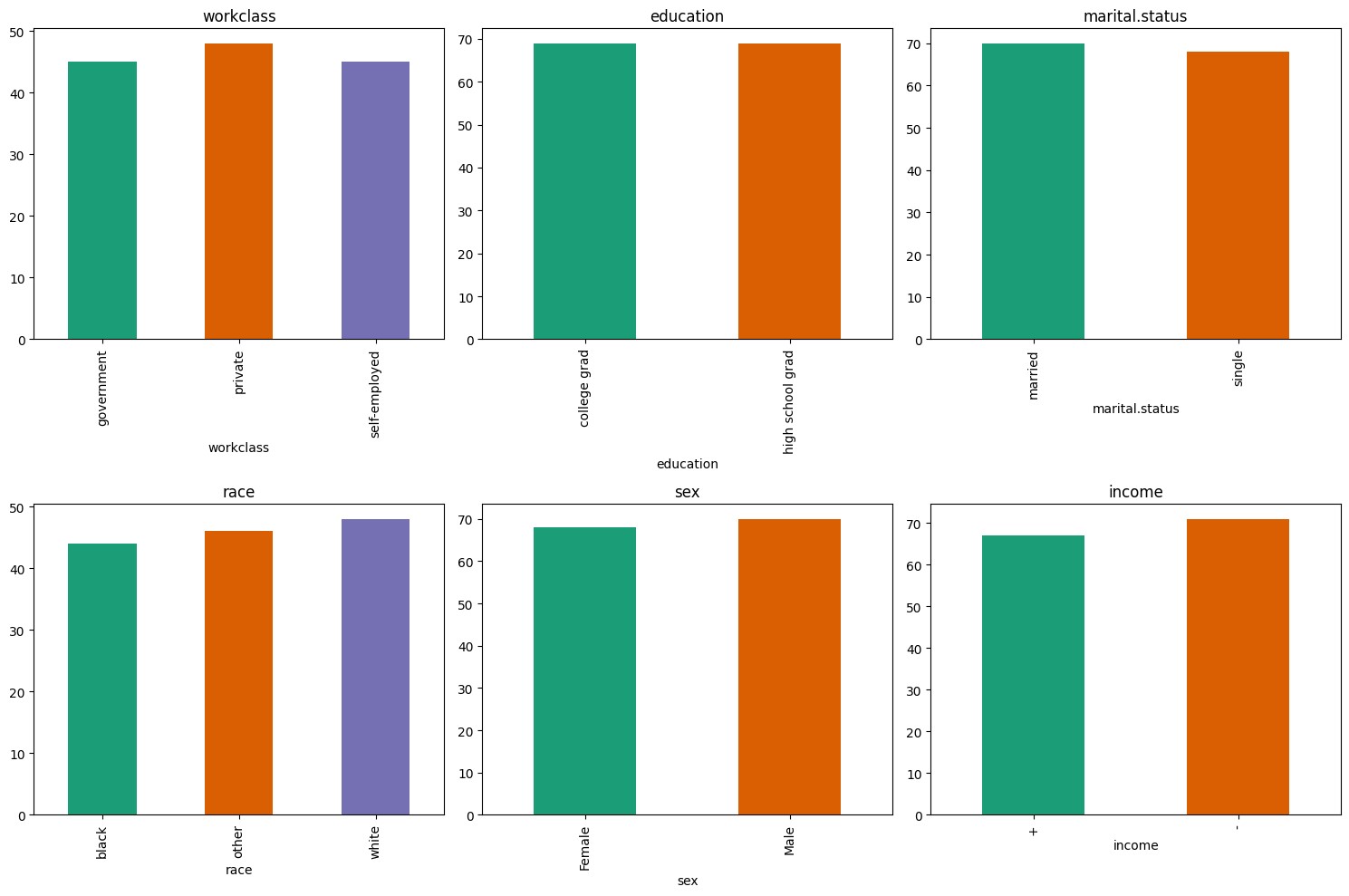
Number of duplicates removed: 30261

Number of remaining samples: 1995

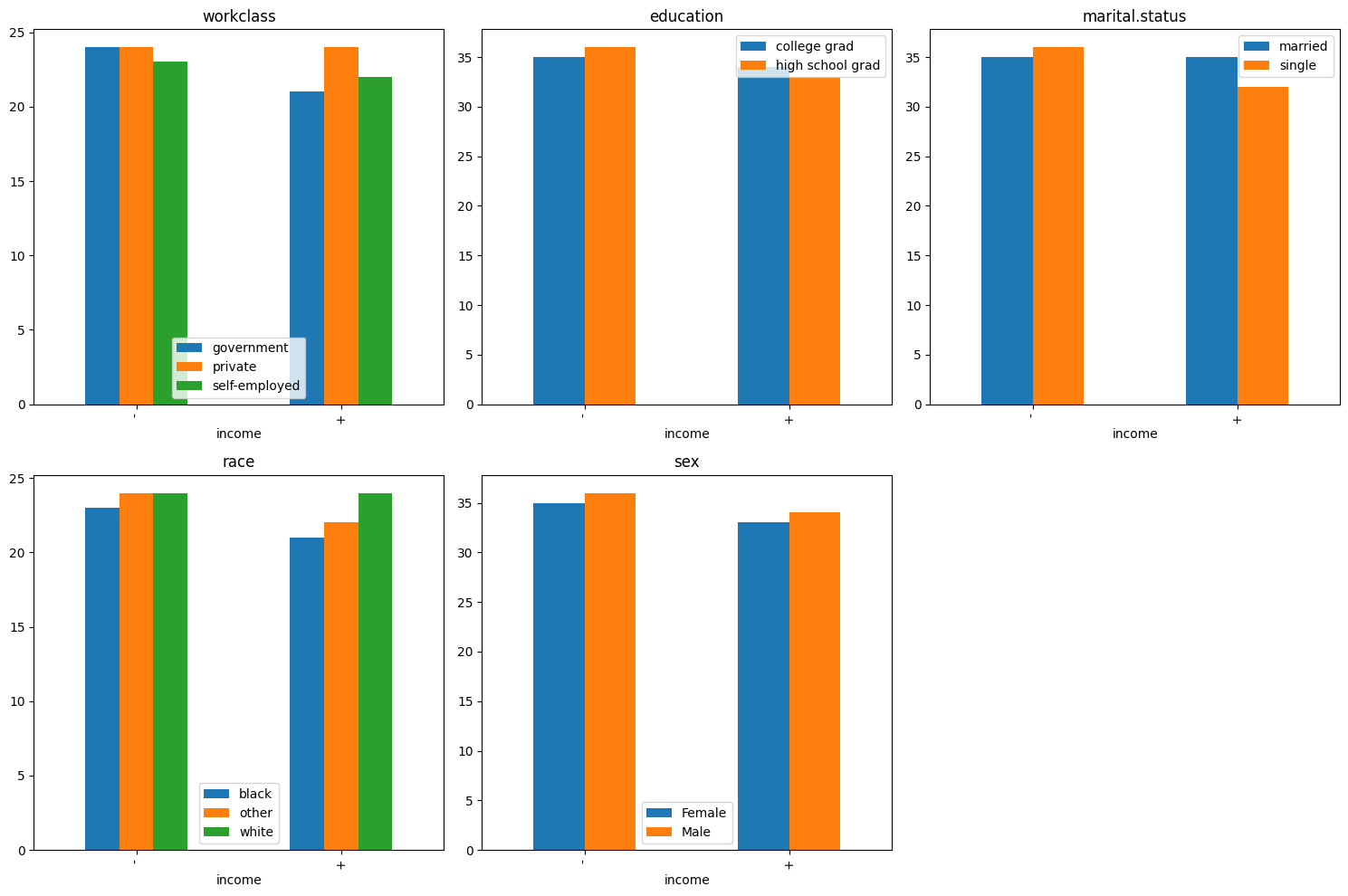
Plots before apply mapping:

Income distribution per each feature before mapping:



Plots after mapping:

Income distribution after mapping:



Find-S h:

array(['?', 'college grad', 'single', '?', '?'], dtype='<U12')

Candidate elimination version space:

Specific:

['?', 'college grad', 'single', '?', '?']

General:

[['?', 'college grad', '?', '?', '?']]