Predicting the

Productivity of Garment Workers

Team Members:

Ethan Cheung, Moises Jabbour

10/7/2024

Table of Contents

[**Part 1 - Statement/Project Goal 3**](#_imw1hmjbxs15)

[**Part 2 - Dataset Description 3**](#_wmpykzqdoxuh)

[**Part 3 - Data Preprocessing 5**](#_km7jqkjjrugg)

[**Part 4 - Attribute Selection 7**](#_u3og5ux26y68)

[**Part 5 - Results and Analysis 11**](#_9972xuikklo1)

[**Part 6 - Conclusion 24**](#_5g0k4wfwpmru)

[**Part 7 - Team Members and Tasks Performed 25**](#_b2i1fnrxbe66)

[**Part 8 - Appendix and Sources 2**](#_kbubrbdeq8q)**8**

## Part 1 - Statement/Project Goal

The garments industry is recognized as one of the largest industries in the world, with around 60 million people worldwide. It serves a crucial role in the economy of developing countries, typically located in south and southeast Asia. Thus, it is important to keep track of the factors that influence worker productivity and performance so that scientists can analyze the types of environments that maximize the efficiency of employees. Scientists Abdullah Al Imran and Md Shamsur Rahim have performed data mining to source this information, which we will be using to create this project. We aim to predict the productivity of workers in the garment industry, analyze its results, and compare the effectiveness of different models and attribute selection techniques in performing this task.

## Part 2 - Dataset Description

The dataset contains 14 attributes, with “actual productivity” as its class. The description of each attribute is provided below.

| Date | Date in MM-DD-YYYY |
| --- | --- |
| Quarter | The corresponding quarter of the month, with each month being split into 4 quarters |
| Department | Associated department of the instance (sewing or finishing) |
| Day | Day of the week |
| Team | Associated team number with the instance (1-12) |
| Target productivity | Worker productivity assigned by the supervisor (assuming 1 means perfect level production and 0 meaning no production) |
| SMV | Standard minute value, time allocated to complete a task (basic time + allowance) |
| WIP | Work in progress, # of unfinished products |
| Overtime | Amount of overtime by each team in minutes |
| Incentive | Amount of financial incentive (BDT) to complete a task |
| Idle time | Amount of time production was interrupted |
| Idle men | Amount of men idle due to production interruption |
| No\_of\_style\_change | Number of changes in a particular product’s style |
| No\_of\_workers | Number of workers in each team |

For further elaboration, the formula for basic time in the SMV calculation is (observed time\*speed of operation)/100, and relaxation allowance + contingency allowance + machine delay allowance for allowance. This dataset contains a numerical continuous class that predicts the general productivity of the day. As such, we may need to perform discretization or create a new class with its information, such as using the target productivity attribute to create a class to determine whether the productivity had been met.

Possible Use Cases

The dataset could have many impactful effects across industries that manufacture garments, and possibly even other industries as some factors may have proportional effects in different fields. For managers, the information from the dataset could help optimize their work by identifying factors that create productive workers and the environments that would support them. By creating models to analyze productivity patterns, resources can be allocated more effectively, training programs can be optimized, and processes can also be expedited. The dataset could produce insights that could help companies meet deadlines and avoid overworking employees, overall likely leading to worker satisfaction and retention.

The dataset could also help garment businesses identify potential issues, such as excessive workloads or unsafe conditions. This could also improve the timing of fixes and the facilitation of changes in the workplace. The data could also establish more transparency about practices, which would help consumers view the garment industry as more trustworthy.

## Part 3 - Data Preprocessing

Preprocessing was done in Weka.

Part 3.1 - Handling Missing Values

To handle the missing values, we removed the WIP (# of unfinished products) attribute, as it had 506 missing values, which is 42% of the number of instances. As a significant amount of instances did not have a value for it, we decided that replacing the missing values was going to produce a poor result. Since there were no other attributes with missing values in the dataset, it was a trivial problem.

Part 3.2 - Discretization

With no other missing values, we performed discretization for attributes with over 20 distinct values. This included SMV (standard minute value), overtime, bonus incentives, idle men, idle time, and the number of workers. The attributes were binned into 10, 14, 5, binarized into 2, 2 again, and 6 ranges of numerical values with even frequency respectively. This was done with Weka’s discretization function, with the even frequency variable switched to True. As there was a large majority of instances with no idle men or idle time, we decided to discretize them into two bins, effectively binarizing them into whether there were idle men or idle time at all during the day the instance was collected.

Part 3.3 - Removing Redundant Attributes

To remove the unnecessary attributes, we looked through each attribute to determine whether the two attributes were related or did not have valuable information for creating our models. We identified Quarter (the work year quarter the data was recorded on) as a redundant attribute, as it was derivable from date. We thus removed the attribute Date, as Quarter is akin to the discretized version of Date and could have some predicting power in demonstrating worker productivity over time. Through this, we ensured that there would be no remaining redundancy in our attributes.

Part 3.4 - Fixing the Class Attribute

We nominalized the class attribute by changing it from productivity into productivity\_met using the attribute targeted\_productivity, with the values yes or no. This was done with the Python script below.

import pandas as pd

csv\_file = 'garments\_worker\_productivity\_classification\_preprocessed.csv'

df = pd.read\_csv(csv\_file)

if 'productivity\_met' in df.columns:

df['productivity\_met'] = df['productivity\_met'].map({1:'yes', 0:'no'})

df.to\_csv(csv\_file, index=False)

Part 3.5 - Data Splitting

To split the datasets into train test validation, we decided to use Python’s pandas and sci-kit-learn libraries to create a script to do so. We decided to use stratified random sampling to keep the same distribution of class attributes among the datasets and chose to use a 70-15-15 split to maintain a healthy balance between dataset sizes. With our five datasets in a directory, the script creates a new directory with a folder for each of the datasets containing its train test and validation sets. The code is shown below.

import os

import pandas as pd

from sklearn.model\_selection import train\_test\_split

input\_dir = "Attribute Selection Datasets"

output\_dir = "Splitted Datasets"

dataset\_class = 'productivity\_met'

os.makedirs(output\_dir, exist\_ok=True)

for filename in os.listdir(input\_dir):

if filename.endswith(".csv"):

file\_path = os.path.join(input\_dir, filename)

df = pd.read\_csv(file\_path)

train\_df, temp\_df = train\_test\_split(

df,

test\_size=0.30,

stratify=df[dataset\_class],

random\_state=42

)

val\_df, test\_df = train\_test\_split(

temp\_df,

test\_size=0.50,

stratify=temp\_df[dataset\_class],

random\_state=42

)

dataset\_directory = os.path.join(output\_dir, filename[:-4])

os.makedirs(dataset\_directory, exist\_ok=True)

train\_df.to\_csv(os.path.join(dataset\_directory, "train.csv"), index=False)

val\_df.to\_csv(os.path.join(dataset\_directory, "validation.csv"), index=False)

test\_df.to\_csv(os.path.join(dataset\_directory, "test.csv"), index=False)

## 

## Part 4 - Attribute Selection

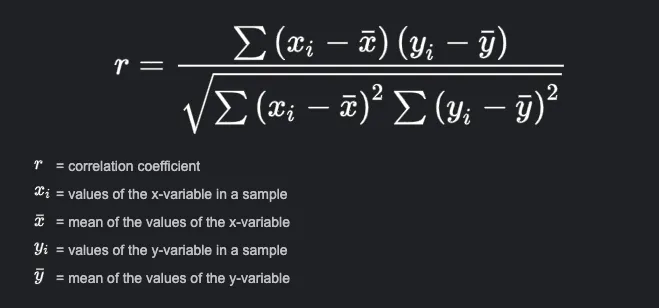
Attribute Selection was done in Weka and Python.

Part 4.1 - Attribute Selection Algorithms

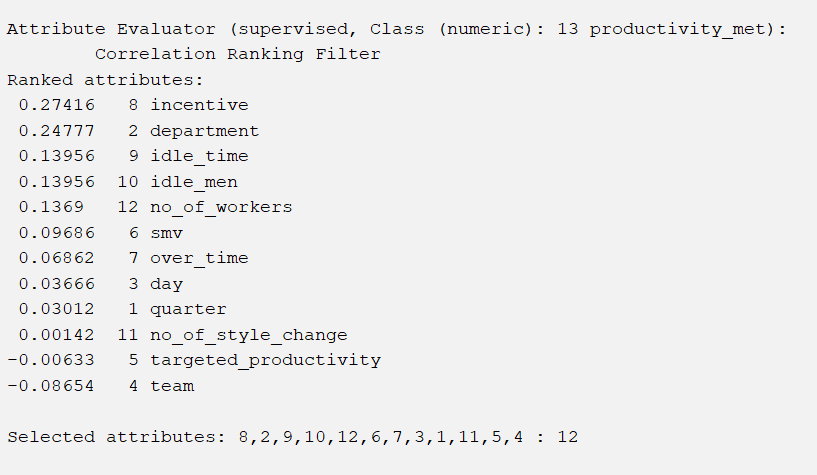
After properly cleaning and formatting the dataset, we can perform attribute selection methods in Weka.

1. **CorrelationAttributeEval**

The CorrelationAtributeEbal measures the correlation between each attribute and the class. It does this by computing the Pearson’s correlation coefficient defined as such:



Where x0 -xn represents the values of a given attribute and y0-yn represents the values of the class and represents the mean on the attribute and represents the mean of the class. The resulting r value ranges from -1 to 1.

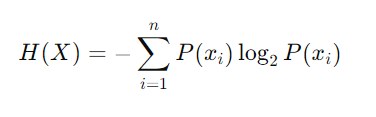


Using a cutoff value of 0.03333, we removed the attributes quarter, no\_of\_style\_change, and targeted\_productivity as they were calculated to have a low correlation with the class attribute.

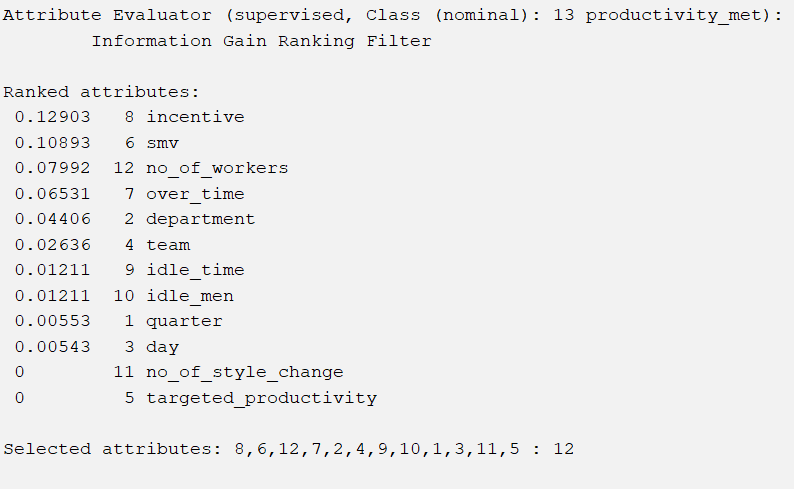
1. **InfoGainAttributeEval**

The InfoGainAttributeEval evaluates the value of an attribute by measuring the information gained concerning the class. Information Gain (IG) is entropy or the measurement of uncertainty in the dataset. Information gain is measured using these formulas:





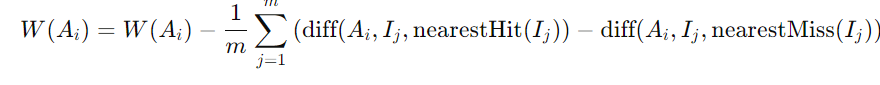
Where H(Class) is the entropy of the class before knowing the attribute and H(Class|Attribute) is the entropy of the class given the attribute where xi represents the label of a particular value in the dataset.



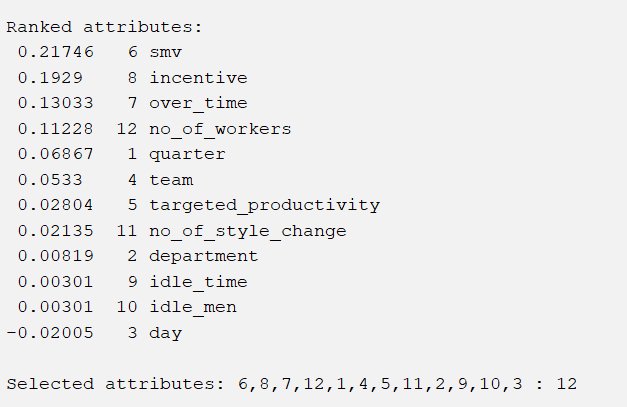
We decided to remove the attributes day, no\_of\_style\_change, and targeted\_productivity as they were shown to have the lowest information gain in our dataset. The ranking showed similarity with CorrelationAttributeEval, as no\_of\_style\_change, and targeted\_productivity were both ranked the worst.

1. **ReliefFAttributeEval**

The ReliefAtributeEval evaluates attributes by estimating how well they distinguish between instances that are near each other. It does this by considering the differences between instances of the same label(nearest hit) and those of a different label(nearestMiss) This is calculated by this formula:



Where m is the number of instances, diff(A, Ij, Ik) is the difference between the label Ij and label Ik of the same attribute. Attributes with higher weights are considered more relevant for distinguishing between labels.



With this, we decided to remove day, idle\_time, and idle\_men as they were shown to be irrelevant to the ability to distinguish between instances of different classes.

1. **WrapperSubsetEval with the J48 Decision Tree**

The WrapperSubsetEval evaluates attribute subsets using specific machine learning algorithms, particularly the J48 Decision Tree. This evaluation uses the performance of the J48 classifier to determine the effectiveness of a subset of attributes in making predictions. The J48 Decision Tree constructs a tree using information gain (see above) to divide the dataset into distinct attributes. This process continues until all instances are classified into a single class and no attributes remain. Additionally, the algorithm employs pruning, which removes branches that contribute little to the algorithm's accuracy. The result of running this algorithm was to remove the team, no\_of\_style\_change, idle\_time, and idle\_men attributes, creating our fourth dataset.

1. **Intuition**

For the fifth dataset, we chose attributes based on our intuitions. We first decided to remove the quarter attribute, as it cannot be changed in the real world if it is shown to have good predicting power. The same applied to ‘day’, i.e. Monday, Tuesday, and so forth, as the worker's mood depending on the day couldn’t be changed either. Lastly, we removed idle time and idle men because they were largely uncontrollable and non-frequent.

Part 4.2 - Classifier Models

1. **1-R**

OneR or “One Rule” is a simple classification algorithm that generates a rule based on a single attribute to classify the data. It works by checking each attribute for how well it performs in classifying data, it then picks the best-performing attribute.

1. **J48**

J48 is a decision tree algorithm that is an extension of 1-R. It makes a set of rules based on multiple attributes, it does this by looking at the information gained from each attribute(as seen in the attribute selection section) to divide the dataset into distinct attributes. This process continues until all instances are classified into a single class and no attributes remain. Additionally, the algorithm employs pruning, which removes branches that contribute little to the algorithm's accuracy.

1. **Random Forest**

Random Forest constructs multiple decision trees and outputs the mode of the classes from all the trees. Each tree is built using a random subset of the training data and a random number of features.

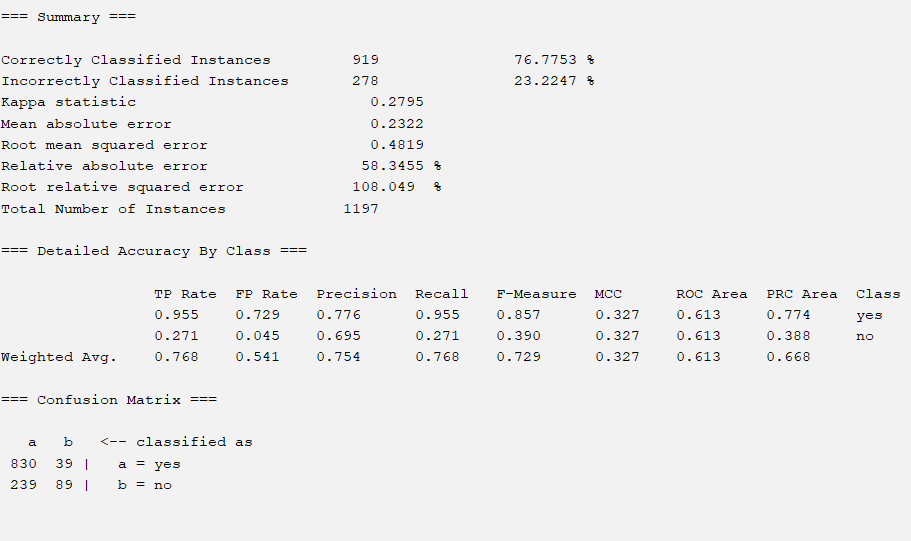
1. **Decision Table**

It is a simple classification model that places instances in a table format. It then creates a set of rules based on combinations of attributes for each row. For classification, it performs a look-up on a table and selects the closest matching row.

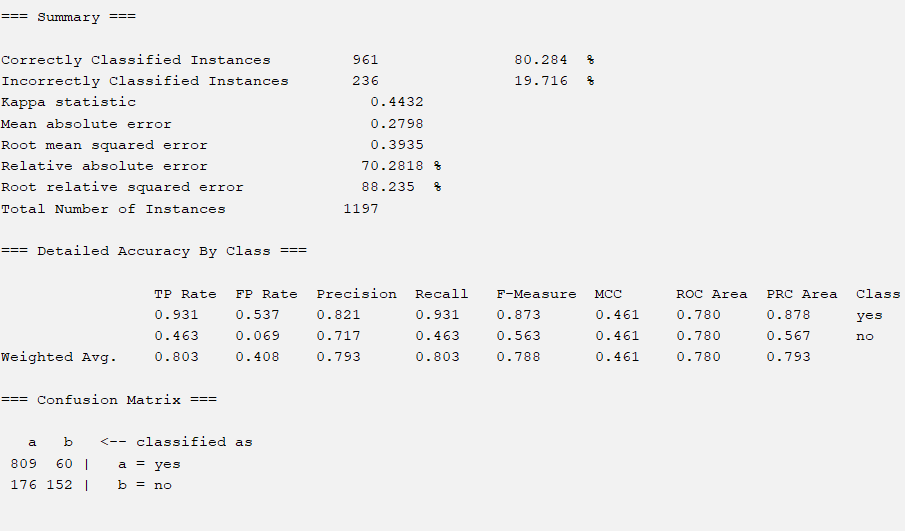
## Part 5 - Results and Analysis

Part 5.1 - Results

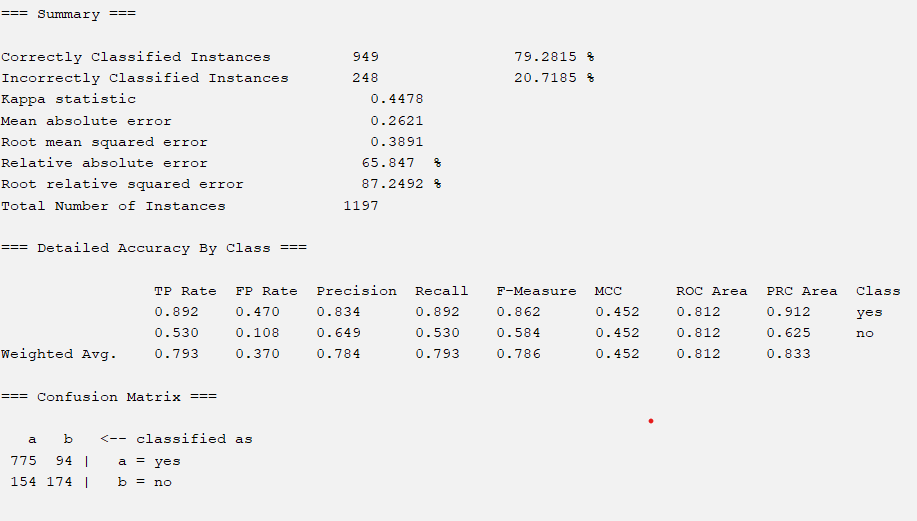
CorrelationAttributeEval with OneR



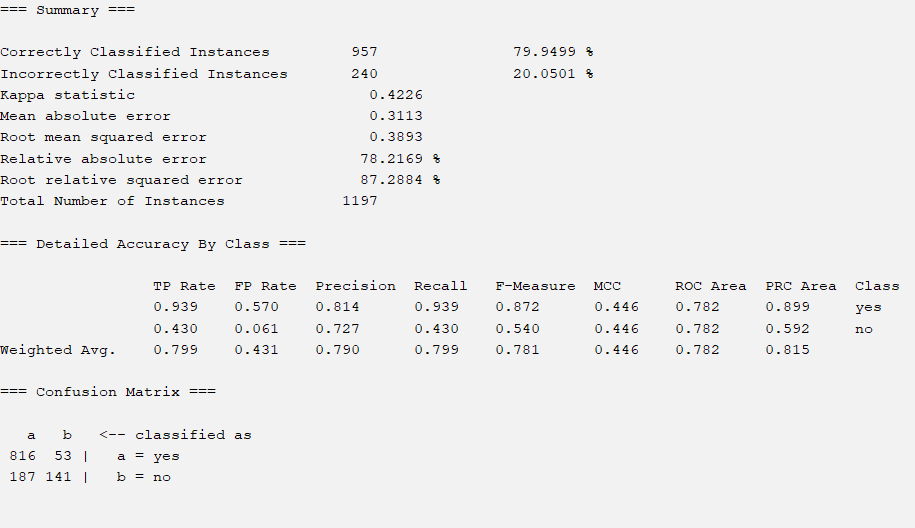
CorrelationAttributeEval with J48



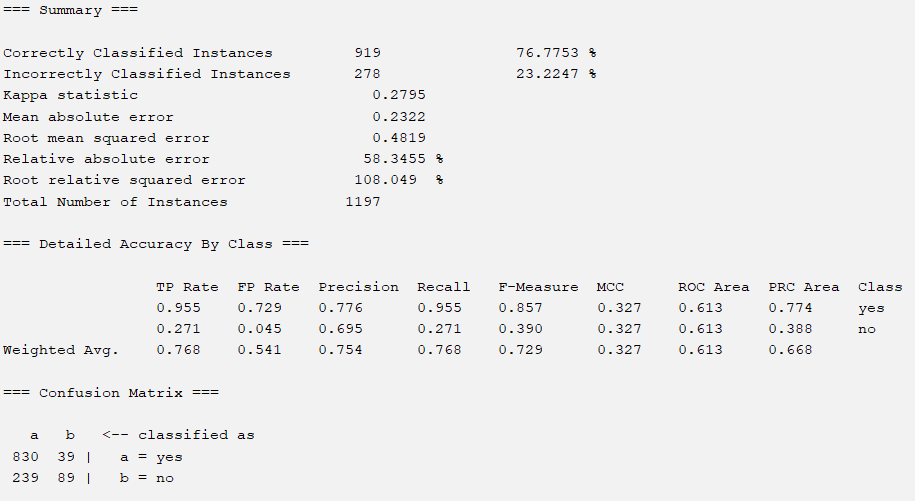
CorrelationAttributeEval with Random Forest



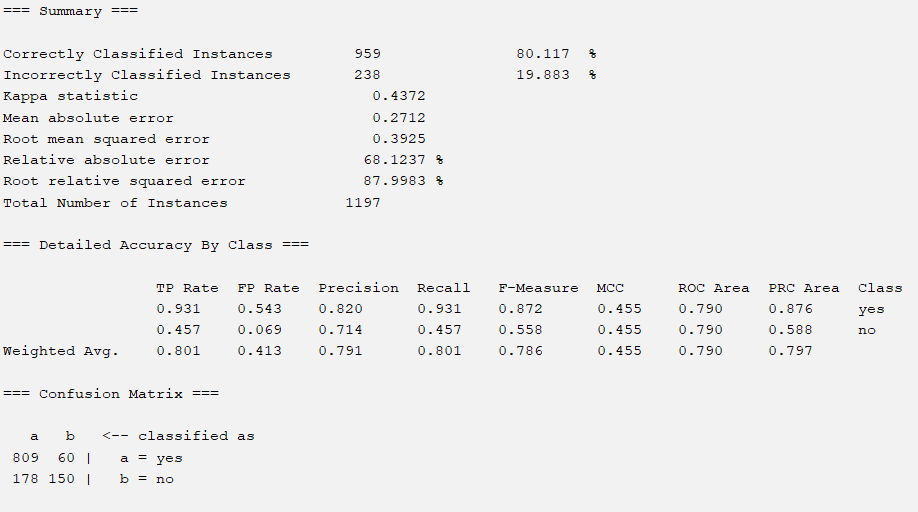
CorrelationAttributeEval with Decision Table



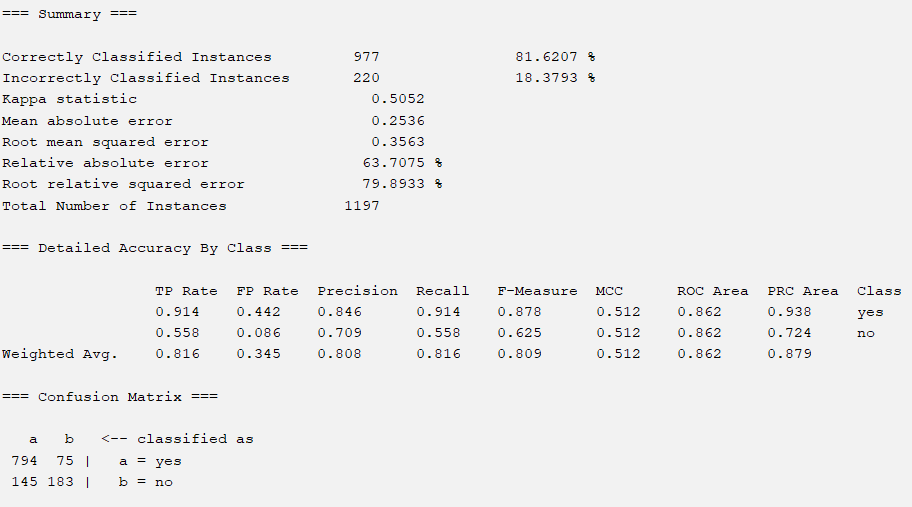
InfoGainAttributeEval with OneR



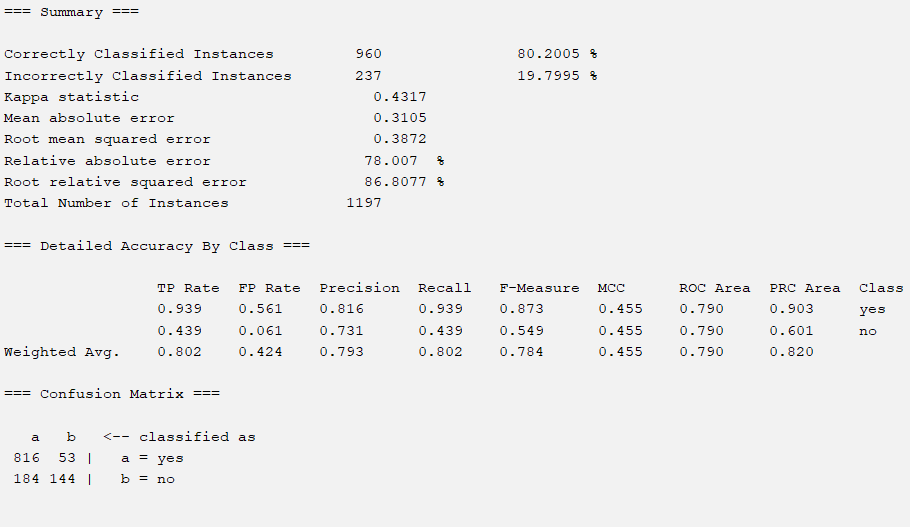
InfoGainAttributeEval with J48



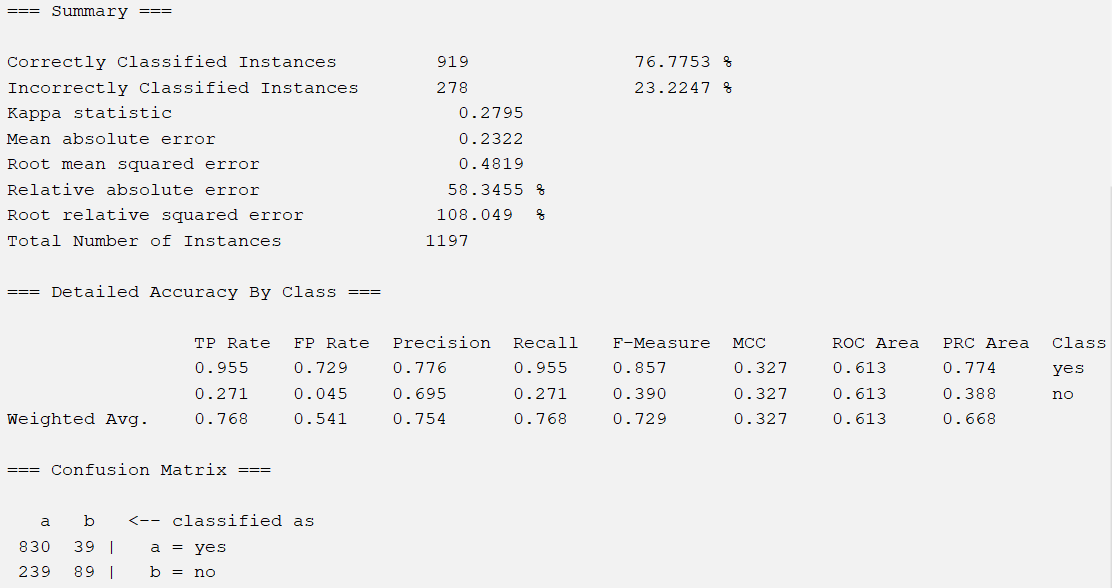
InfoGainAttributeEval with Random Forest



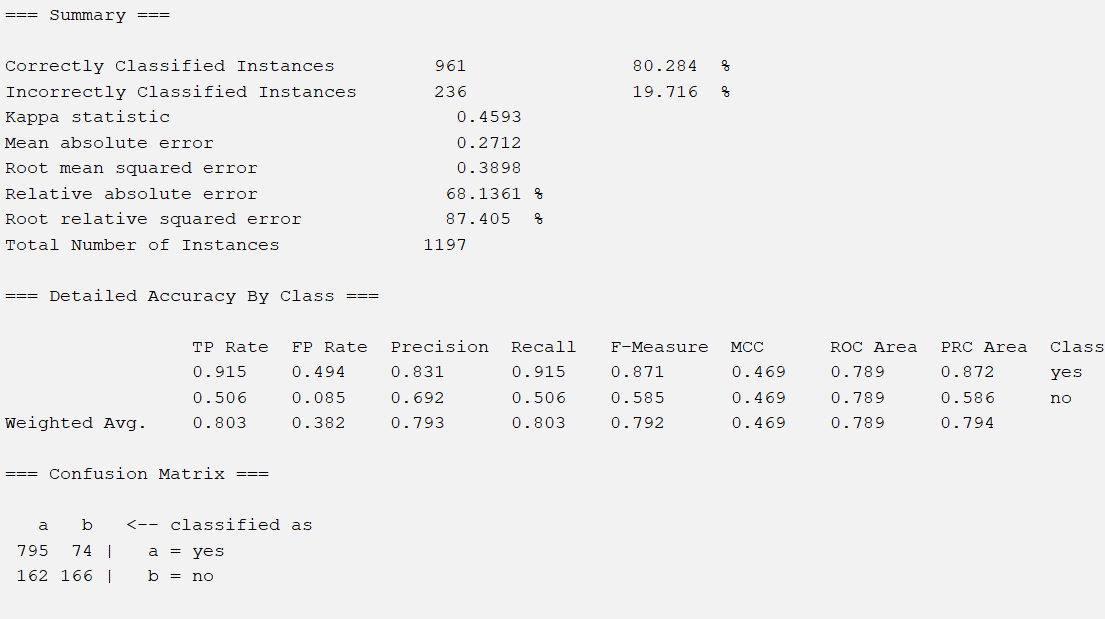
InfoGainAttributeEval with Decision Table



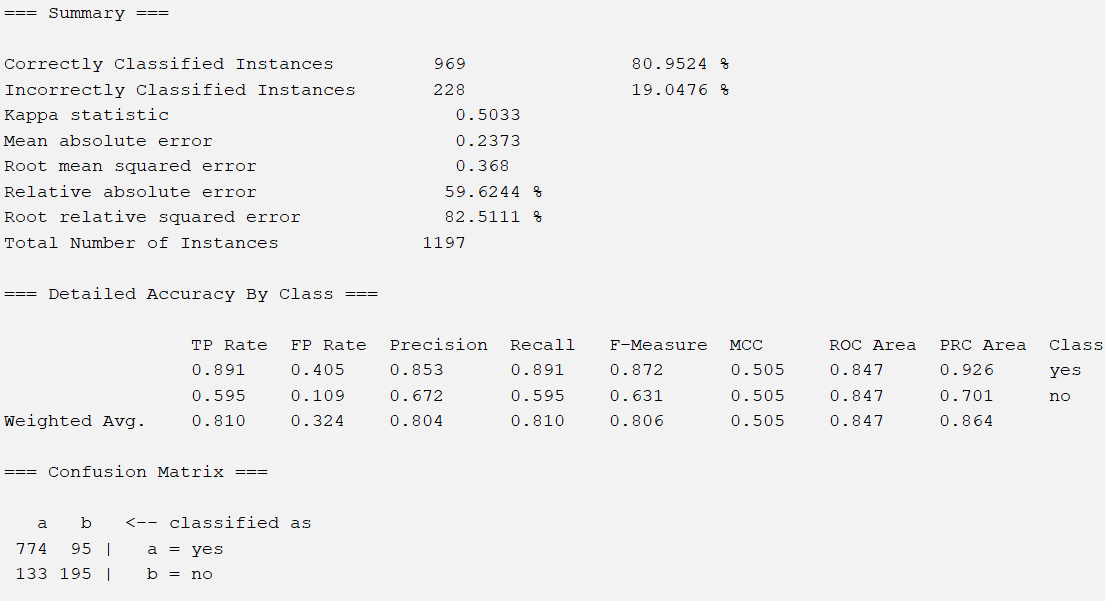
ReliefFAttributeEval with OneR



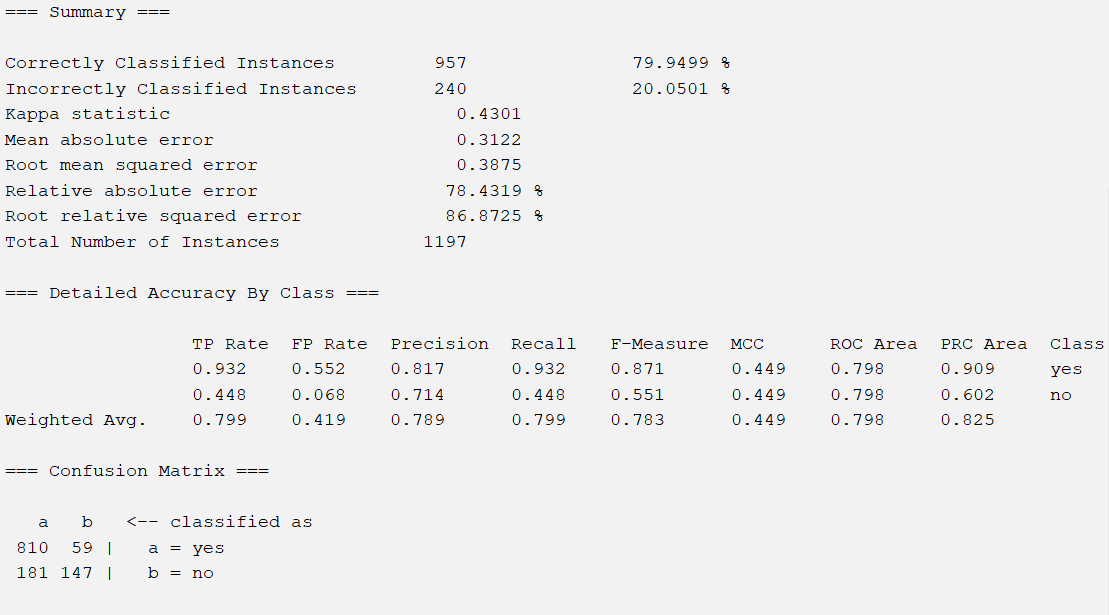
ReliefFAttributeEval with J48



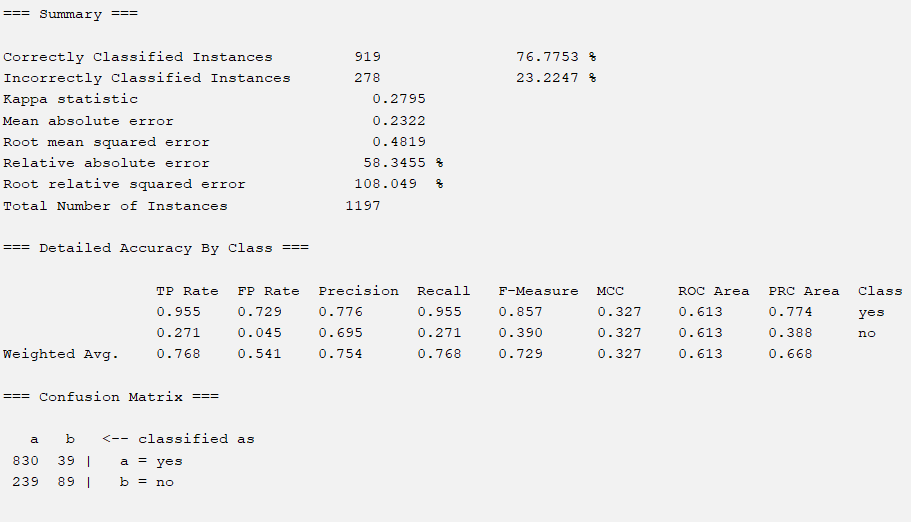
ReliefFAttributeEval with Random Forest



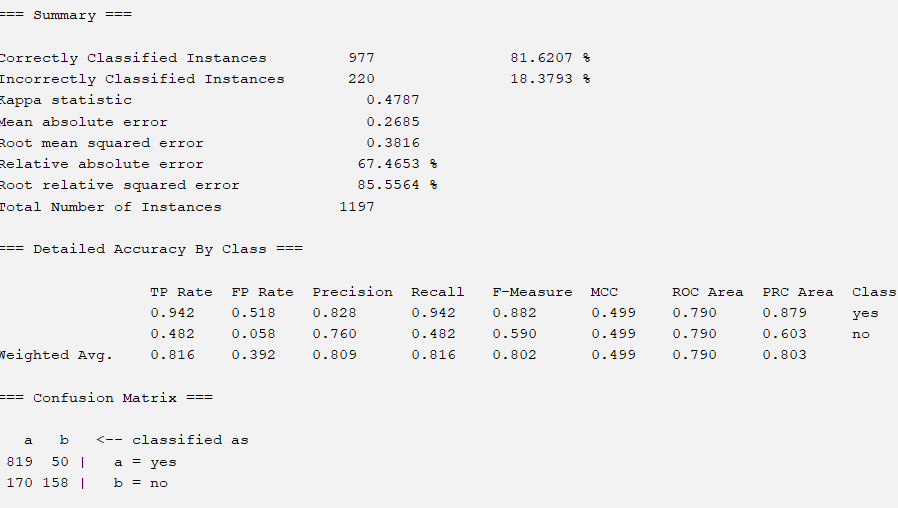
ReliefFAttributeEval with Decision Table



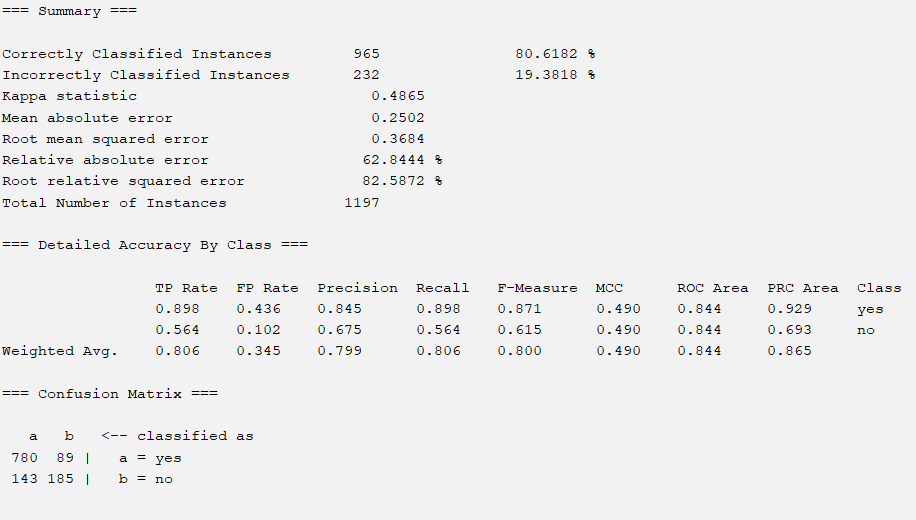
WrapperSubsetEval with OneR



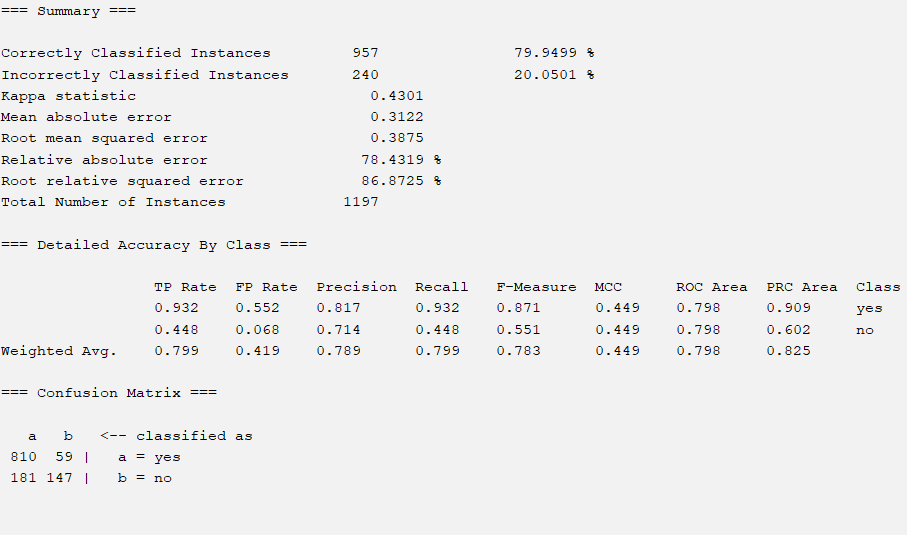
WrapperSubsetEval with J48



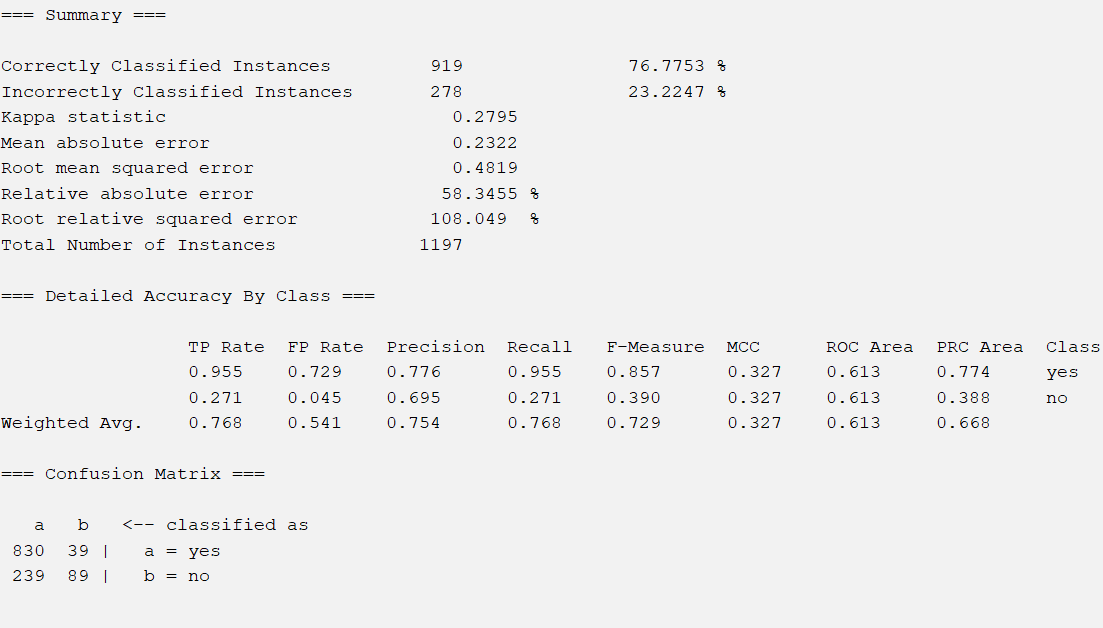
WrapperSubsetEval with Random Forest



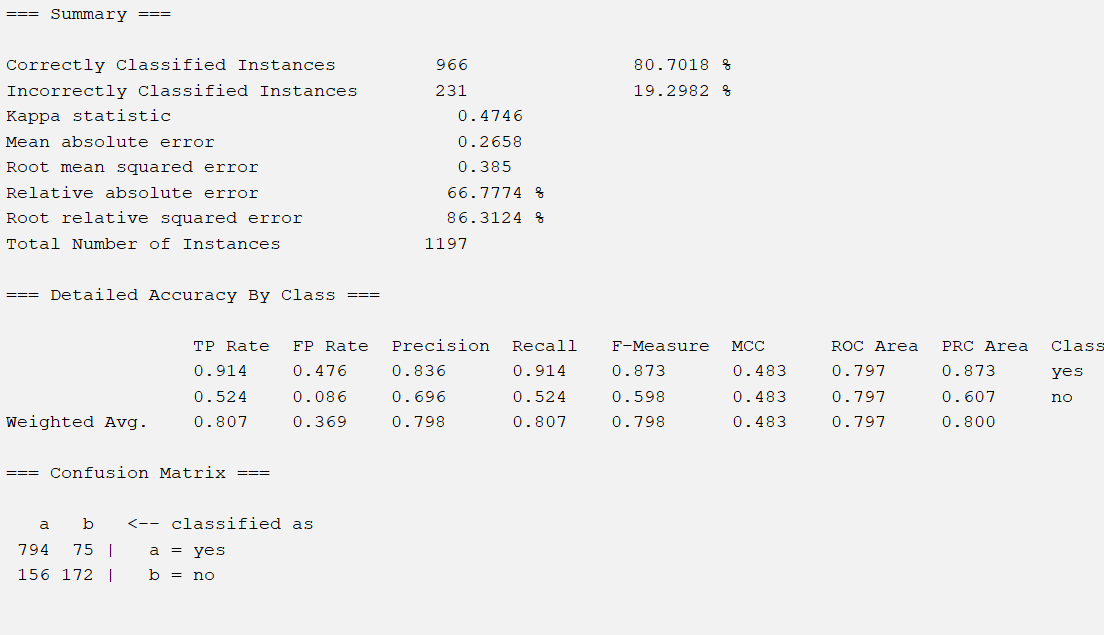
WrapperSubsetEval with Decision Table



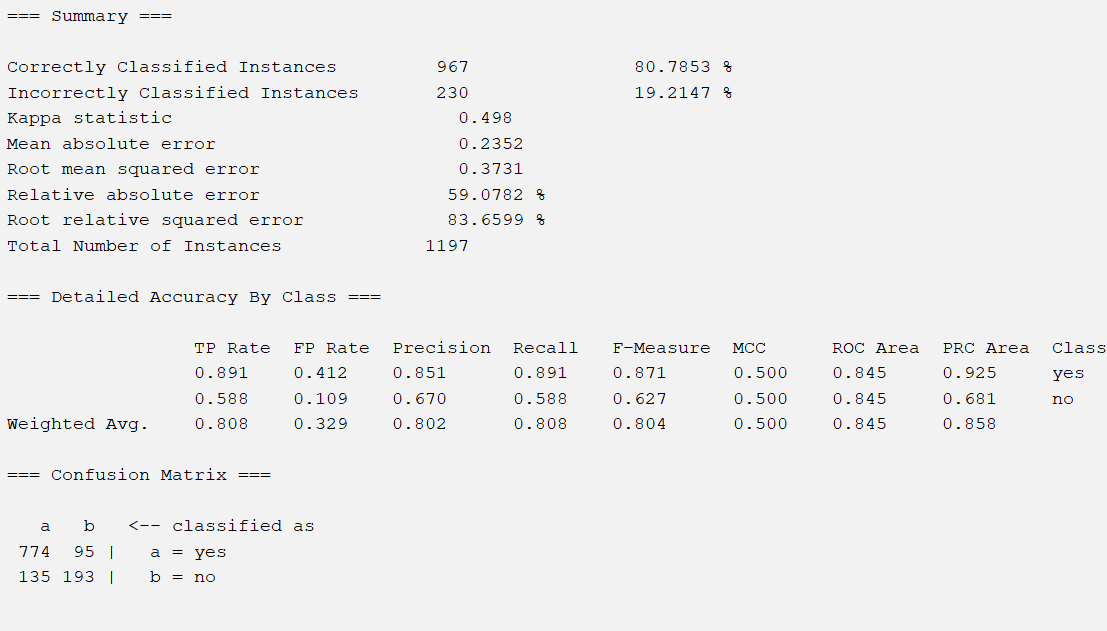
Intuition with OneR



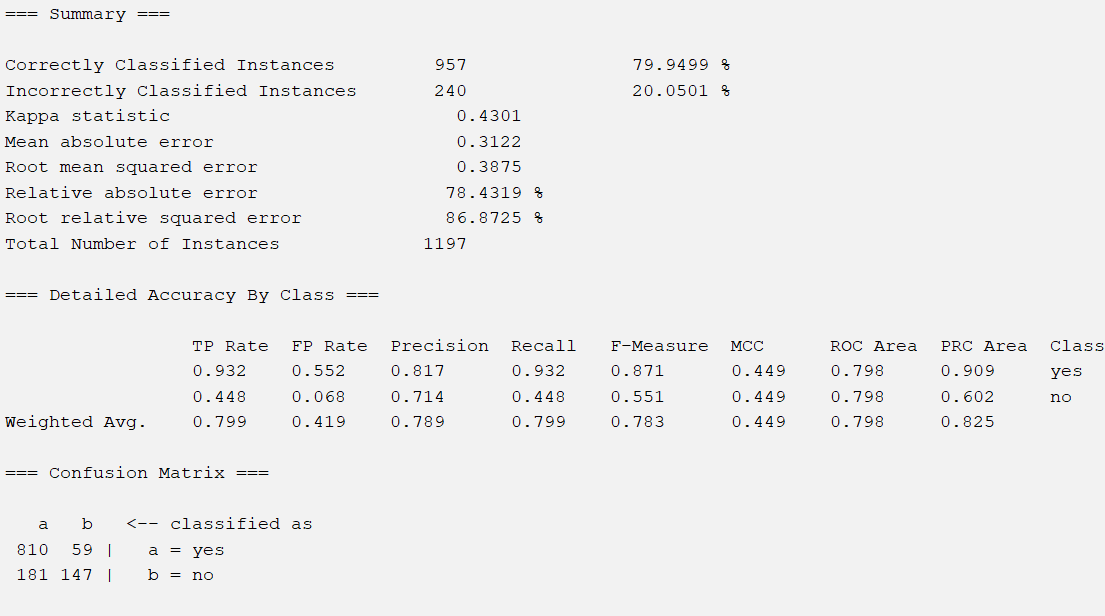
Intuition with J48



Intuition with Random Forest



Intuition with Decision Table



Accuracy Comparison

|  | **Correlation** | **Info Gain** | **Relief** | **WrapperSubset** | **Intuition** |
| --- | --- | --- | --- | --- | --- |
| **1-R** | 76.78% | 76.78% | 76.78% | 76.78% | 76.78% |
| **J48** | 80.28% | 80.12% | 80.28% | 81.62% | 80.70% |
| **Random Forest** | 79.28% | 81.62% | 80.95% | 80.62% | 80.79% |
| **Decision Table** | 79.95% | 80.20% | 79.95% | 79.95% | 79.95% |

Confusion Matrices Comparison

|  | **Correlation** | **Info Gain** | **Relief** | **WrapperSubset** | **Intuition** |
| --- | --- | --- | --- | --- | --- |
| **1-R** | | 830 | 39 | | --- | --- | | 239 | 89 | | | 830 | 39 | | --- | --- | | 239 | 89 | | | 830 | 39 | | --- | --- | | 239 | 89 | | | 830 | 39 | | --- | --- | | 239 | 89 | | | 830 | 39 | | --- | --- | | 239 | 89 | |
| **J48** | | 809 | 60 | | --- | --- | | 176 | 152 | | | 809 | 60 | | --- | --- | | 178 | 150 | | | 795 | 74 | | --- | --- | | 162 | 166 | | | 819 | 50 | | --- | --- | | 170 | 158 | | | 794 | 75 | | --- | --- | | 156 | 172 | |
| **Random Forest** | | 775 | 94 | | --- | --- | | 154 | 174 | | | 794 | 75 | | --- | --- | | 145 | 183 | | | 774 | 95 | | --- | --- | | 133 | 195 | | | 780 | 89 | | --- | --- | | 143 | 185 | | | 774 | 95 | | --- | --- | | 135 | 193 | |
| **Decision Table** | | 816 | 53 | | --- | --- | | 187 | 141 | | | 816 | 53 | | --- | --- | | 184 | 144 | | | 810 | 59 | | --- | --- | | 181 | 147 | | | 810 | 59 | | --- | --- | | 181 | 147 | | | 810 | 59 | | --- | --- | | 181 | 147 | |

3543

Part 5.2 - Analysis

After running each of our four models on the five datasets created through different attribute selection techniques, we found general accuracy to be around 80%. The following results had the highest performance:

InfoGainAttributeEval with Random Forest

WrapperSubsetEval with J48

ReliefFAttributeEval with Random Forest

Intuition with Random Forest

Intuition with J48

With a tie for first, we would select the chosen model to be either **InfoGainAttributeEval with Random Forest** or **WrapperSubsetEval with J48**. They both had an accuracy of 81.62%, with InfoGainAttributeEval with Random Forest handling false negatives better but false positives worse, with 183/328 with negatives and 794/869 positives compared to 158/328 and 819/869. With some uncertainty on how to proceed, we turn to the four variations of error scores: mean absolute error, root mean squared error, relative absolute error, and root relative squared error.

Across the four, we found that **InfoGainAttributeEval with Random Forest** had the lowest error outputs. Thus, we can be assured that if we were to run the model on future datasets the precision, recall, and accuracy will remain consistent with little variations in performance.

Although it would be optimal to build a model that achieves higher accuracy, the nature of the dataset we selected makes it rather difficult to do with the simple models listed above. A deep neural network would likely perform better in performing these classification tasks. However, being able to predict productivity roughly 80% of the time and understanding the factors that help achieve that could still serve significant uses to garment manufacturers across the world. The results demonstrate that there is some correlation between these factors and productivity, which is significant in further optimizing the processes that fuel our world. In the future, we could look deeper into the model to understand how each attribute was weighted, but for now, instead, we may look back at what the attribute selection algorithms had outputted to conclude.

| **Correlation** | **Info. Gain** | **Relief** | **Wrapper** |
| --- | --- | --- | --- |
| 0.27416 8 incentive  0.24777 2 department  0.13956 9 idle\_time  0.13956 10 idle\_men  0.1369 12 no\_of\_workers  0.09686 6 smv  0.08654 4 team  0.06862 7 over\_time  0.03666 3 day  0.03012 1 quarter  0.00633 5 targeted\_productivity  0.00142 11 no\_of\_style\_change | 0.12903 8 incentive  0.10893 6 smv  0.07992 12 no\_of\_workers  0.06531 7 over\_time  0.04406 2 department  0.02636 4 team  0.01211 9 idle\_time  0.01211 10 idle\_men  0.00553 1 quarter  0.00543 3 day  0 11 no\_of\_style\_change  0 5 targeted\_productivity | 0.21746 6 smv  0.1929 8 incentive  0.13033 7 over\_time  0.11228 12 no\_of\_workers  0.06867 1 quarter  0.0533 4 team  0.02804 5 targeted\_productivity  0.02135 11 no\_of\_style\_change  0.00819 2 department  0.00301 9 idle\_time  0.00301 10 idle\_men  -0.02005 3 day | Selected Attributes:  quarter  department  Day  targeted\_productivity  smv  over\_time  incentive  no\_of\_workers  productivity\_met |

In these 4 groups, we can see that attributes 8, 6, 2, 9, 12, 7, and 4 all tend to rank high in similar order. These attributes are explained in **Part 1 - Statement/Project Goal**.

Overall, attributes such as incentive and standard minute value were demonstrated to have the highest importance, which would make sense as they are likely to heavily influence how workers are motivated. Attributes such as the number of style changes, targeted\_productivity, or day all relatively rank low. As these realistically do not affect workers, it would make sense that they are not valued by the attribute selection algorithms. However, among each attribute, it is hard to clearly distinguish one that would directly change how workers operate, which might serve as a reason why accuracy wasn’t able to reach beyond 82%.

## Part 6 - Conclusion

As aforementioned, our **InfoGainAttributeEval with Random Forest** model had the best results when compared to the rest of the 20 runs in this project. I believe that this project has been largely successful, as a model able to determine whether productivity was met roughly 80% of the time could have significant implications for the garment industry. Future projects could be conducted to further analyze each of the models, allowing us to examine specific attribute weights and whatnot. It would also be of interest to compare results with a more complex model, such as a deep neural network.

Steps to Reproduce Our InfoGainAttributeEval with Random Forest Model:

1. Run nominalize\_class.py on Weka and load Garment\_worker\_productivity\_clasification.csv
2. Open Weka and load the new dataset
3. Remove WIP and Date attributes in the preprocess tab
4. Still, under the PreProcess tab, click Filter > Choose > filters > unsupervised > attribute, and select Discretize.
5. Enable useEqualFrequency to True
6. Apply for each attribute as such: discretize SMV (standard minute value), overtime, bonus incentives, idle men, idle time, and the number of workers into 10, 14, 5, 2, 2, 6 bins respectively, modifying attributeIndices which correspond to the attribute accordingly
7. Go to the “Select attributes” tab and choose the correct class – productivity\_met
8. Select InfoGainAttributeEval as the Attribute Evaluator and Ranker as the Search Method and hit Start
9. Keep account of attributes with scores lower than 0.01 and remove them on the preprocess tab
10. Save the dataset as a CSV
11. **The above can be found as Attribute Selection Datasets/garments\_worker\_productivity\_classification\_infogain.csv**
12. Open Weka Explorer and load the dataset on the preprocess tab
13. Click on the Classify tab and click “Cross-validation” under Test Options. Leave folds as 10
14. Under Classifier, select Choose > weka > classifiers > trees > RandomForest
15. Click Start

## Part 7 - Team Members and Tasks Performed

Team Members: Ethan Cheung and Moises Jabbour

Finding the Data & Building Proposal: Ethan Cheung

Preprocessing Initial Attempt: Ethan Cheung

Preprocessing & Project Update: Ethan Cheung

Non-Weka Attribute Selection Algorithm: Ethan Cheung

Attribute Selection Algorithms and Classifiers: Ethan Cheung & Moises Jabbour

Results Output: Moises Jabbour

Results Analysis: Ethan Cheung

Building Final Report: Ethan Cheung & Moises Jabbour

Team Morale: Moises Jabbour

## Part 8 - Appendix and Sources

Data source website: [https://archive.ics.uci.edu/dataset/597/productivity+pr…](https://archive.ics.uci.edu/dataset/597/productivity+prediction+of+garment+employees)

**Files Attached with the Report:**

* Train\_test\_val\_split.py – file used to split the data after attribute selection into training testing and validation sets
* Garment\_worker\_productivity\_clasification.csv– dataset used for preprocessing with the class nominalized
* Garment\_worker\_productivity\_clasification\_preprosses.csv– dataset after preprocessing
* Nominalize\_class.py– file used to change the class from a continuous variable to a nominalized class

| **Table (Back to File Summary)** | Garment Worker Productivity Dataset(Raw) |
| --- | --- |
| **Description** | Raw CSV of the dataset |
| **File Name** | Garment\_worker\_productivity\_clasification.csv |
| **Data Type** | **Column Name - CSV** |
| Char | Date |
| Char | Quarter |
| Char | Department |
| Char | Day |
| Int | Team |
| Float | Target productivity |
| Float | SMV |
| Int | WIP |
| Int | Overtime |
| Int | Incentive |
| Int | Idle time |
| Int | Idle men |
| Int | No\_of\_style\_change |
| Boolean | No\_of\_workers |

| **Table (Back to File Summary)** | Garment Worker Productivity Dataset(Preprocessed) |
| --- | --- |
| **Description** | Preprocessed CSV of the dataset |
| **File Name** | Garment\_worker\_productivity\_clasification\_preprossesed.csv |
| **Data Type** | **Column Name - CSV** |
| Char | Quarter |
| Char | Department |
| Char | Day |
| Int | Team |
| Float | Target productivity |
| Float | SMV |
| Int | Overtime |
| Int | Incentive |
| Int | Idle time |
| Int | Idle men |
| Int | No\_of\_style\_change |
| Char | No\_of\_workers |

Works Cited

Al Imran, Abdullah, Md Shamsur Rahim, and Tanvir Ahmed. "Mining the Productivity Data of the Garment Industry." *International Journal of Business Intelligence and Data Mining*, vol. 19, no. 3, 2021, pp. 319–342. Inderscience Enterprises Ltd.

Hasan, Mohammad Shahadat, et al. "An Approach to Predict Employee Productivity Using a Hybrid CNN-LSTM Model." *2021 International Conference on Artificial Intelligence (ICAI)*, IEEE, 2021, pp. 1-7,<https://ieeexplore.ieee.org/document/9689701>. Accessed 20 Oct. 2024.

Schlattmann, Maximilian. "OneR: Machine Learning in Under One Minute." *Ephorie Blog*, 13 Mar. 2020,<https://blog.ephorie.de/oner-machine-learning-in-under-one-minute>. Accessed 20 Oct. 2024.

Frank, Eibe, et al. *The WEKA Workbench*. Version 3.8, University of Waikato, 2023, [https://www.weka.io/?utm\_medium=cpc&utm\_](https://ml.cms.waikato.ac.nz/weka) Accessed 20 Oct. 2024.