Assignment #2
Biostat 202

Data Management and Cleaning (due 8/13/2020 at 11:55 PM)

Description: The purpose of this lab is to further familiarize you with Orange Data Mining and, in particular, merging, imputing, and cleaning. We will use data from National Electronic Injury Surveillance System (NEISS) database. The NEISS database is similar to many real-world datasets in that it is messy and requires significant effort to wrangle into a format that permits analysis. Ultimately, we will predict whether a patient was admitted or died as a result of their injury

Skills:

- Import, merge, append data
- Data visualization, filtering, and structural errors
- Missing data and imputation
- Define new features and targets

File: download from CLE

- neiss2018_part1_demographics.csv
- neiss2018_part1_injury.csv
- neiss2018_part2.csv

Steps: *Please follow the steps below and answer questions marked in* ***bold in a separate document****. Include a screenshot of your final workflow.*

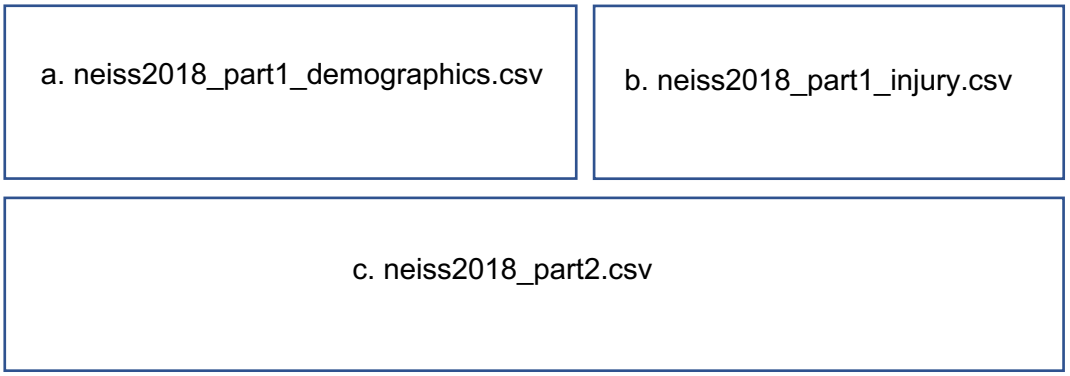
For this homework we will use the data from the National Electronic Injury Surveillance System (NEISS) available at http://www.cpsc.gov/en/Research--Statistics/NEISS-Injury-Data/. The database was designed by the Consumer Product and Safety Commission (CPSC) to monitor consumer products that can cause injuries (think hoverboard). It is a national sample of hospitals in the U.S. that records emergency department visits that involve any sort of injury related to a consumer product. The data from 2018 alone consists of approximately 360,000 records and contains the following variables: CPSC_Case_Number, Treatment_Date, Age, Sex, Race, Other_Race, Body_Part, Diagnosis, Other_Diagnosis, Disposition, Location, Fire_Involvement, Product_1, Product_2, Narrative_1, Narrative_2, Stratum, PSU, and Weight. We will examine a subsample to ensure your computer does not freeze due to the size.

The Disposition variable is coded as 1= treated and released, 2 = transferred, 4 = admitted to hospital, 5 = held for observation, 6 = left emergency department, 8 = died, 9 = not recorded. **We will be interested in predicting if the patient was either admitted (code = 4) or held for observation (code = 5) or died (code = 8).**

Import, merge, append data:

1. Download the three files listed above from the course CLE website.

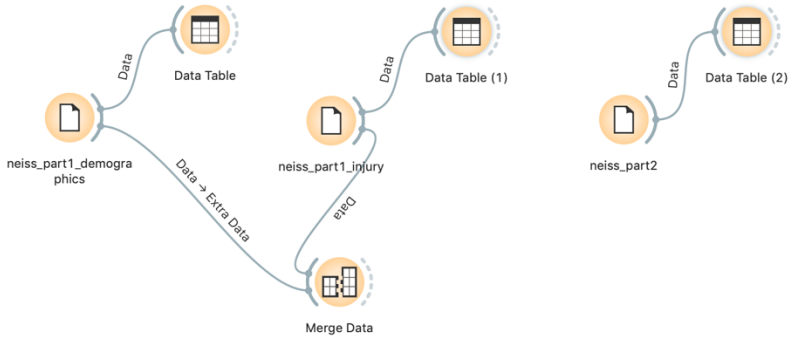
2. We have purposefully partitioned the total NEISS 2018 dataset to provide some exercises in merging the data. The data files are structured as diagrammed below.
    a. neiss2018_part1_demographics.csv: demographics variables – subset1
    b. neiss2018_part1_injury.csv: injury related variables – subset1
    c. neiss2018_part2: demographics and injury variables – subset2

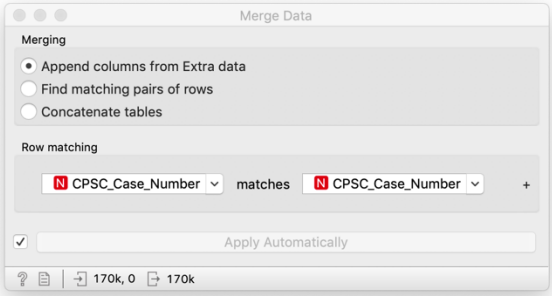


3. Import the three .csv files using separate “File” widgets. Name the widgets by their file source. Keep some space between the separate widgets to ensure you have sufficient room to navigate.

4. Add “Data Table” widgets to each of the files to inspect the contents. **Q1: Which of the three files share common *CPSC_Case_Number* (i.e. patients)?** These two data files share the same subjects but distinct column variables. The remaining dataset contains all column variables but distinct subjects.

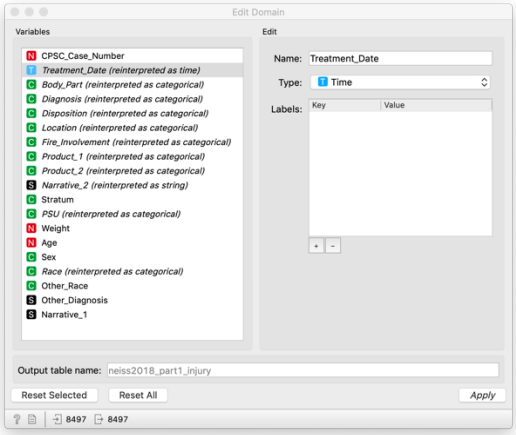
5. Merge by common subjects: We will begin by merging neiss2018_part1_demographics with neiss2018_part1_injury. Select a “Merge Data” widget from the Data tab then feed channels into it from the two relevant “File” widgets. For Row matching select *CPSC_Case_Number*. This means the two data files will be merged with records matched according to the *CPSC_Case_Number* variable. Inspect the results of the merge to confirm the record numbers are correct (8,497 rows). The workflow so far should look like this:



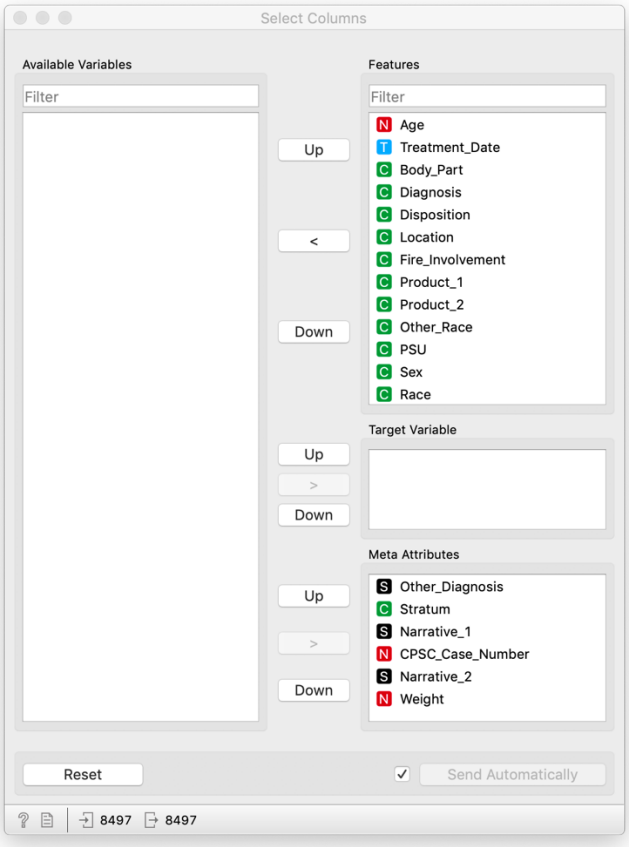


6. Next we will concatenate the newly merged part1 data with the remaining part 2 data. Concatenation means to link together records, often by vertically stacking datasets that share common variables but distinct subjects. Before we can perform a concatenation, we need to confirm that the variables in both datasets share common Types and Roles. Recall, Orange tries to guess these functions as best possible but there may be disagreement which can disrupt the concatenation.

7. To confirm the types agree, add an “Edit Domain” widget to each the “Merge Data” widget and the neiss_part2 “File” widget. Make sure the variable types match and correct any disagreement by modifying the Type via the dropdown menu within the “Edit Domain” widget. Make sure to set *Treatment_Date* as a time variable! The variable types should look as below (most should be categorical given that they represent discrete characteristics rather than continuous ranges). Note that Diagnosis, Location, Product_1/Product_2 are all numeric codes for different diagnoses, body parts, locations where injuries occurred, and products (up to two entered) but because they take on so many values, Orange thinks they are continuous. Fire_Involement is coded for visits that involve fire with 0 = no fire or flame/smoke spread, 1=fire department involved, 2=fire department not involved and 3 = unknown. If you are curious the codebook is on CLE (NEISS Coding Manual.pdf). Product_1 and Product_2 have *many* different values.

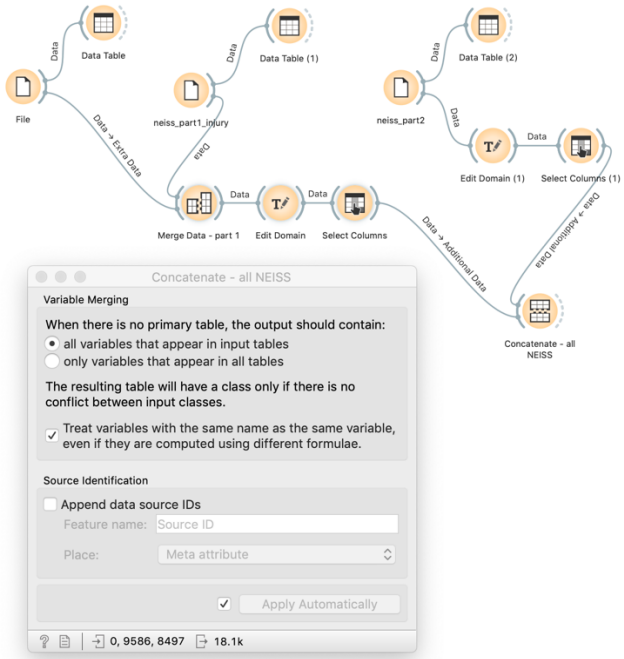


8. Then add a “Select Column” widget and confirm the roles match for each data set. Variable Roles should appear as below.



9. Concatenation: Now that the variable Types and Roles match, we can select the “Concatenation” widget from the Data tab. There are various options you can select to control the concatenation, including restricting the concatenation to variables found in both data sets or only common variables between the two

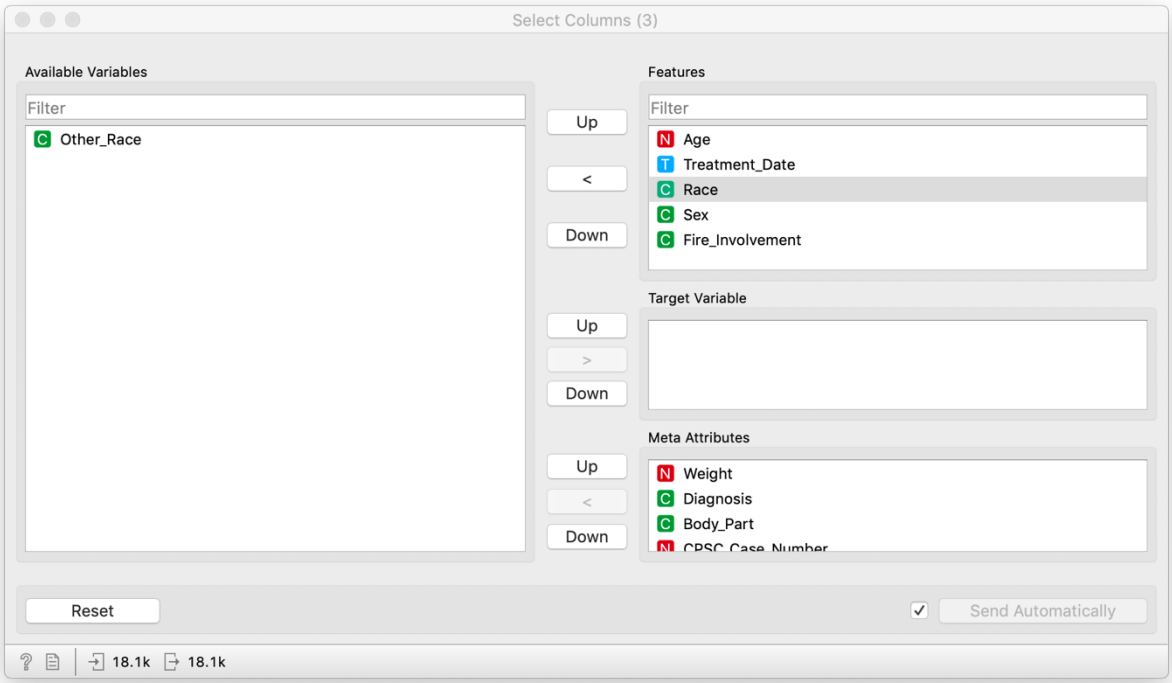
datasets. To avoid losing any data in the event the Types or Roles disagree, I would recommend keeping all variables that appear in input tables. The workflow should look like this:



10. If you wish you can go and rename some of the widgets to clarify the steps taken. We have now imported, merged, and concatenated data.

<u>Data Filtering, Visualization, and Structural Errors:</u>

11. Now that the data set is complete, we will rearrange the variable roles for computational purposes using the “Select Columns” widget again. Orange sometimes struggles when variables with many categories (e.g. *Diagnosis*) so we will make these variables have a role of Meta for the moment until we need to use them. This prevents Orange from attempting to incorporate them into models which may freeze our machine. We will also filter out the *Other_Race* variable since it is almost entirely missing and thus provides almost no information. Your Roles following concatenation should look like this.



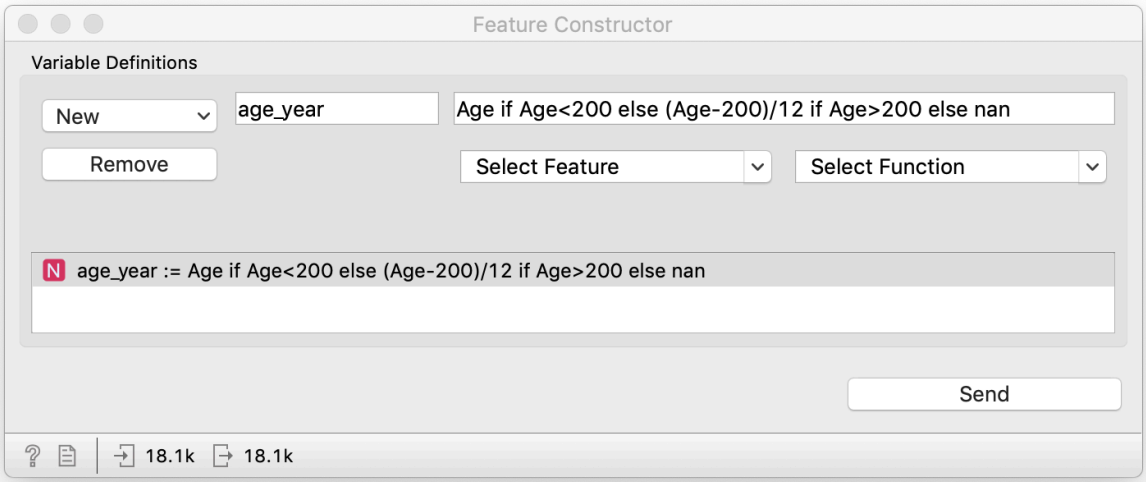
12. Add a “Feature Statistics” widget after you have rearranged the Roles. Notice how the *Treatment_Date* feature is formatted as a Time variable and thus we can measure the center date, dispersion, maximum, etc…The dates are irregular due to how I partitioned the data to reduce the size.

13. Keep examining the “Feature Statistics” widget.
    Q2: Which variables have missing data?
    Q3: Which variables have implausible values? Hint, look at *Age* and *Weight*.
    Q4: Ask yourself, “What are the units for these variables?” How would you confirm the units?

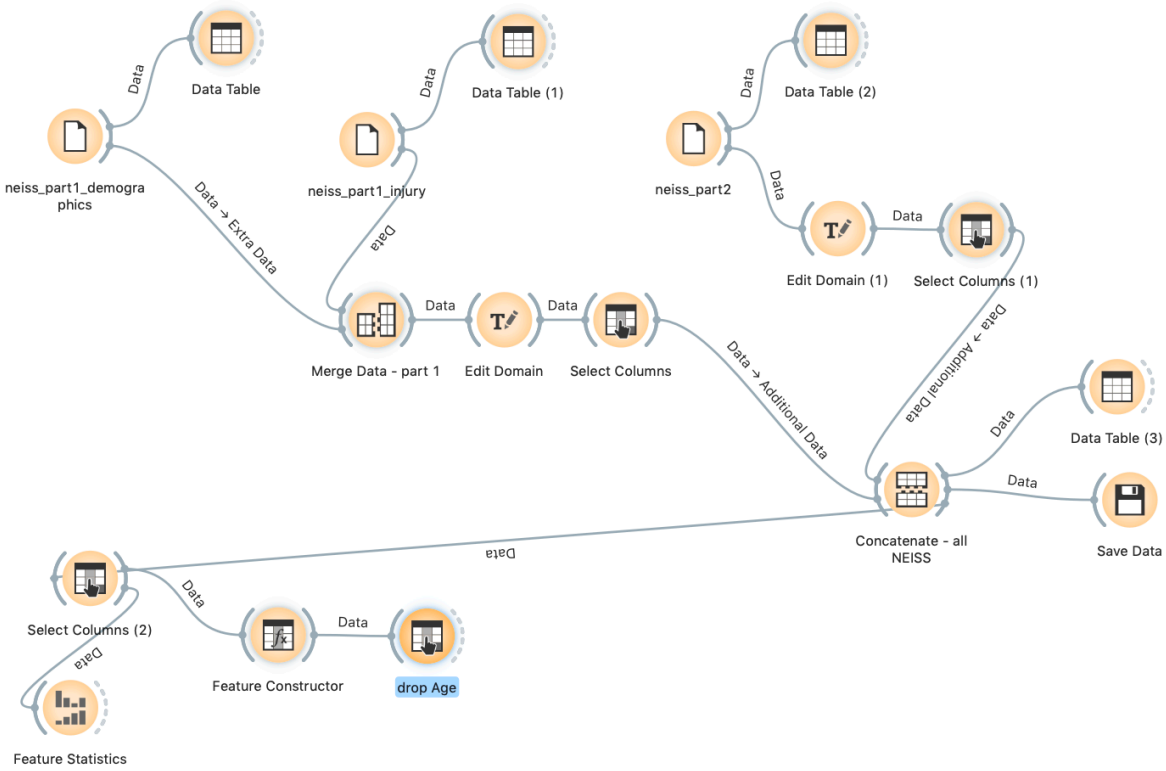
14. Do you think the Weight variable corresponds to patient body weight? In this case the Weight variable corresponds to the probability of the hospital in question being included in the survey. ALWAYS check and confirm the definition of the variables in your data set as they can sometimes be misleading.

15. Age has a maximum value of 223. This seems troubling. According to the NEISS 2019 Codebook, *Age* ‘0’ is an indicator that *Age* was not recorded and thus should be marked as blank or missing. *Age* values with the structure ‘2xx’ denote children under two years old where ‘xx’ is the child’s age in months. However, Orange does not know this and thinks all children are somehow hundreds of years old. To correct this, we will have to use the “Feature Constructor” widget with some conditional logic to create a variable *age_year* which will record age in years for all subjects. Essentially, we will take all values of the form ‘2xx’ and reformat them into years and set all values equal to ‘0’ as missing. Your feature

constructor widget should look as follows (conditional logic: Age if Age<200 else (Age-200)/12 if Age>200 else nan):

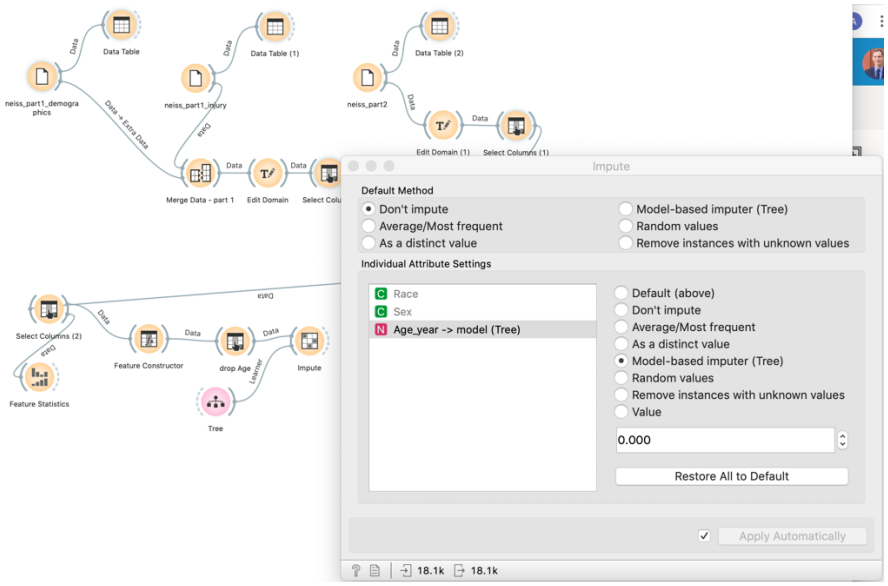


16. Drop the old *Age* variable using “Select Columns” widget. Your workflow should look like this so far:



Missing Data and Imputation

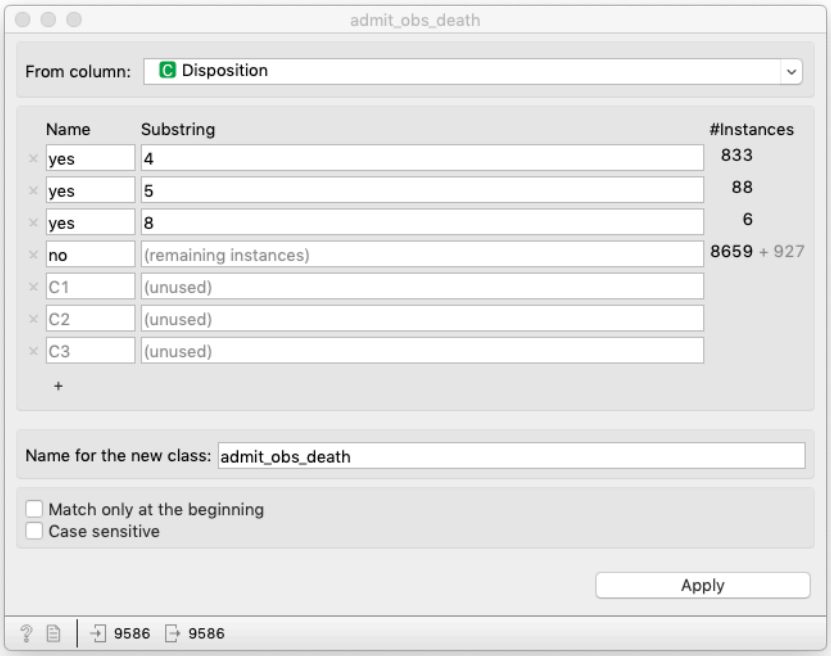
17. *Age_year* will have missing values (both due to '0' values as well as blank values). Therefore, we should fill in these missing values with our best guess. Orange allows you to do this using the "Imputation" widget. Add the "Imputation" widget after you have dropped *Age.* For the default option, select 'Don't Impute'. Then under Individual Attribute Settings select *Age_year* and choose 'Model-based imputer (single tree)'. This uses a decision tree to fill in the missing values for *Age_year*, using the other features as predictors. To verify that a simple tree is used, add a "Tree" widget to the left-hand side of the "Impute" widget. This tells Orange that it will impute the data using a tree. **Q5: Think back to lecture and consult the "Imputation" widget. What are other strategies you could have used to address the missing data?**



18. Add a "Feature Statistic" widget to verify that the missing *Age_year* data has been imputed. Looks like it has!

Define new features and targets:

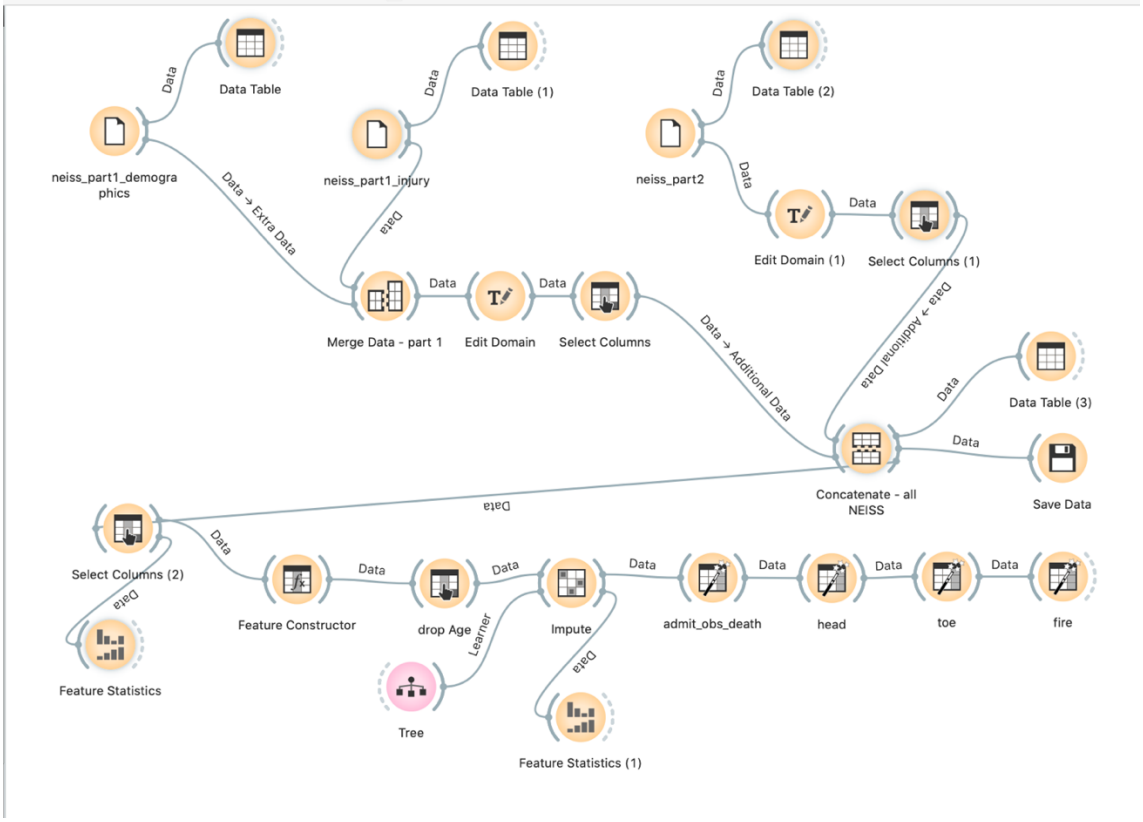
19. We will now create our target variable by reclassifying the *Disposition* variable which we previously mentioned at the beginning of lab. To do so, we will create a categorical variable that has 'yes' patient was either admitted (code = 4) or held for observation (code = 5) or died (code = 8) and 'no' otherwise. Select a "Create Class" widget from the Data tab and connect it to your workflow. For 'From column:' select *Disposition.* Under Name, enter three rows of 'yes' with 4, 5, and 8 in the corresponding entry for Substring. Then enter a 'no' under Name and leave the Substring blank. Finally, enter *admit_obs_died* as the Name for new class. Name this widget "admit_obs_died". Your selection should look like below:



20. Repeat the same process using the “Create Class” widget to create three new variables in addition.
    a. Head involved with injury (*head*): Body_part (yes = 75, no otherwise)
    b. Toe involved with injury (*toe*): Body_part (yes = 93, no otherwise)
    c. Fire involved with injury (*fire)*: Fire_Involvement (yes = 1,2,3, no otherwise)

    Q6: Why do you think it is important to derive new variables like *head* and *toe* when predicting hospital admittance, observation, or death? Why not just include the *Body_part* variable as is?

Your workflow should look roughly like this:



Prediction of target:

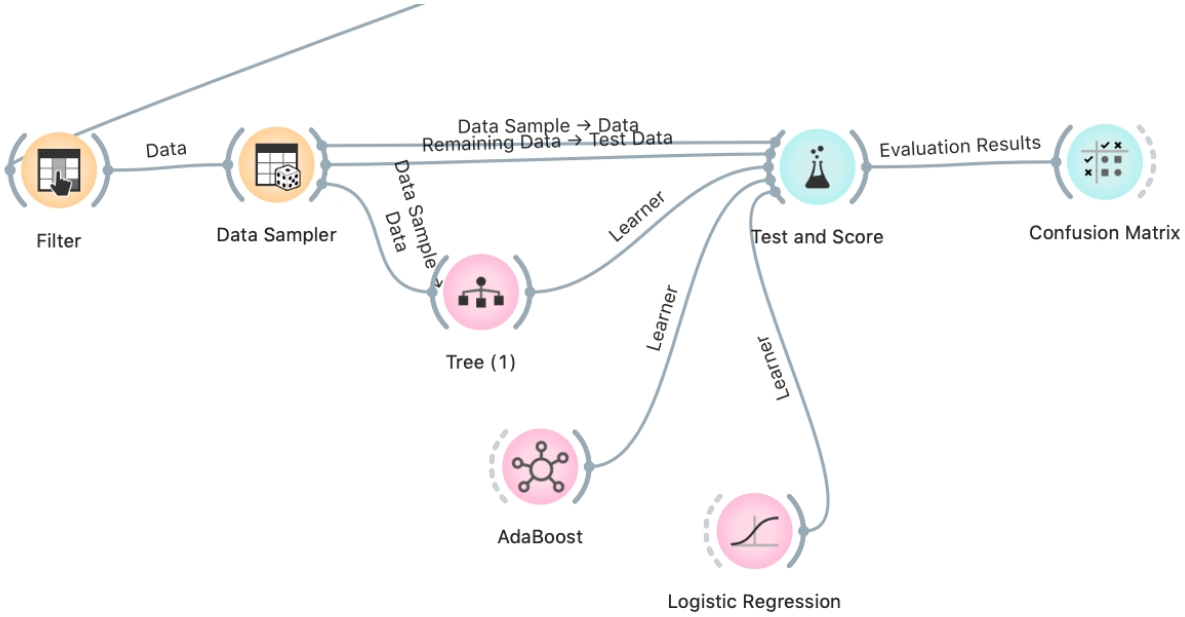
21. We are nearly ready to predict our target, but do so we need to add one last "Select Columns" widget to make sure our variables are sorted properly. We wish to predict the target *admit_obs_death* using features *Race, Sex, Age_year, fire, head, and toe*. All other variables should be filtered or set as Meta. See below:

22. Add a "Data Sampler" widget and divide into 70% training and 30% testing as we did in Assignment 1. Make sure to assign both the training and test data to the "Test and Score" widget in the next step.

23. Add a "Test and Score" widget and select at least three learners that are appropriate for categorical data. See steps in Assignment 1 to configure the "Test and Score" widget with the three learners. Possible learners include:
    a. Tree
    b. kNN
    c. Random Forest
    d. SVM
    e. Logistic Regression
    f. Naïve Bayes
    g. AdaBoost

24. **Q7: Which learner did best in training for accuracy (CA)? In testing? Do you think the model is doing well?**

25. Add a “Confusion Matrix” widget to the right-side of the “Test and Score” widget. This shows you the cross-tabulation of actual vs predicted categories. **Q8: Do you still think the model is doing well? Hint: look at the balance of accurately predicted events within each class of our target variable. Why or why not?** An example of my “Confusion Matrix” is pasted below along with the relevant workflow:



Please include a screenshot of your final workflow. My final workflow is pasted below.