Homework 2: ML Pipeline

Jean Salac (salac@uchicago.edu)

April 11, 2018

GitHub Repo: `https://github.com/jeansalac/ml-ppol`
Collaborator: Yuliana Zamora

1. **Read Data:**

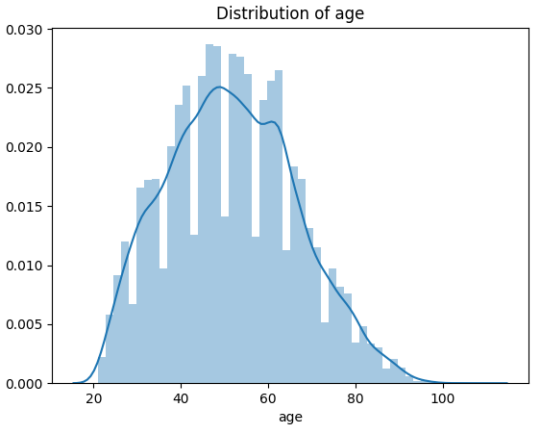
   - To read in the csv file, I wrote a function in my library *pipeLib.py* called that takes in the csv file through command line and converts it to a data frame using the PANDAS library function *read_csv()*.

2. **Explore Data:**

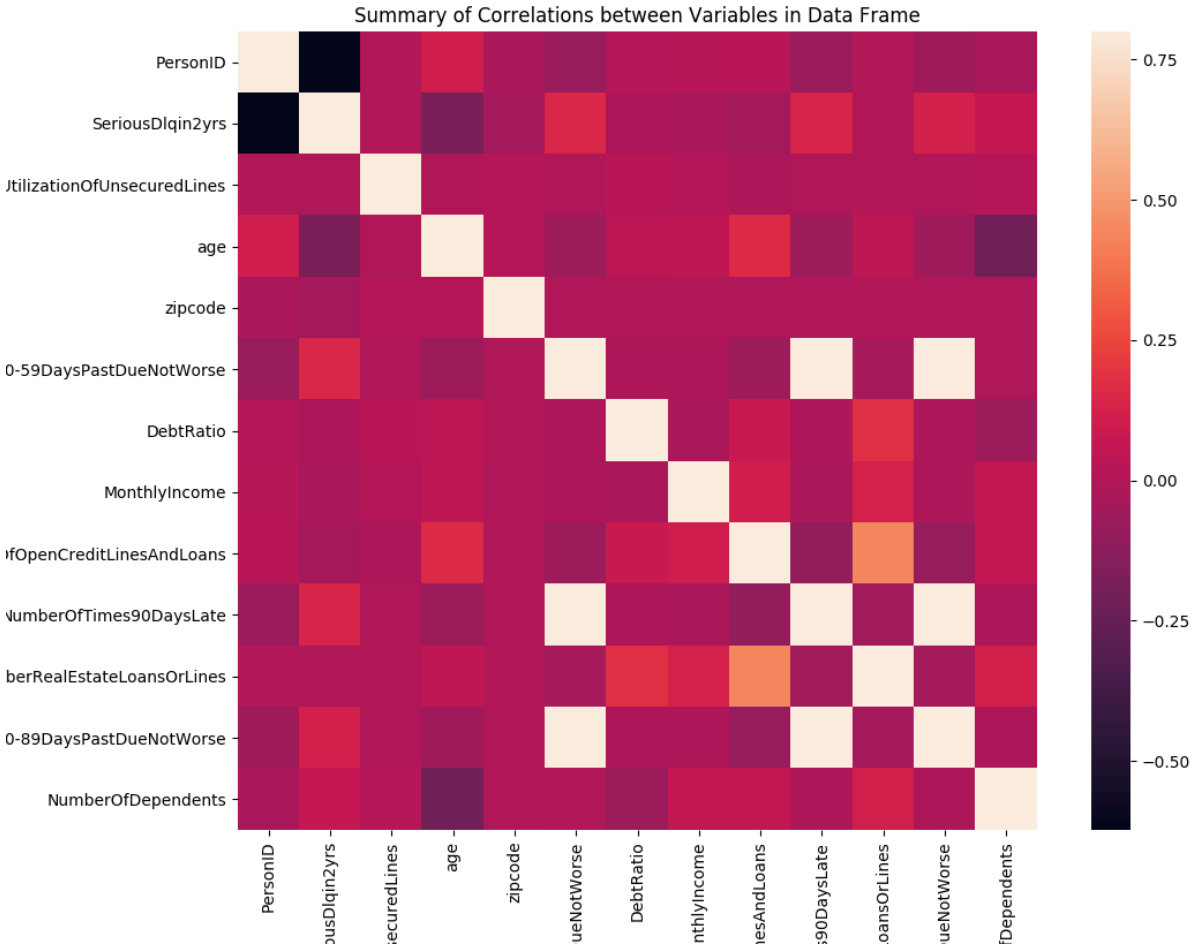
   - *dataSummary(dataFrame,var)*: This function takes in a *dataFrame* object and *var*, the variable you would like a data summary for and returns the summary statistics of *var*. Running this function on the variable *age* in *pipeTest.py* yields:

count	41016.000000
mean	51.683489
std	14.746880
min	21.000000
25%	41.000000
50%	51.000000
75%	62.000000
max	109.000000

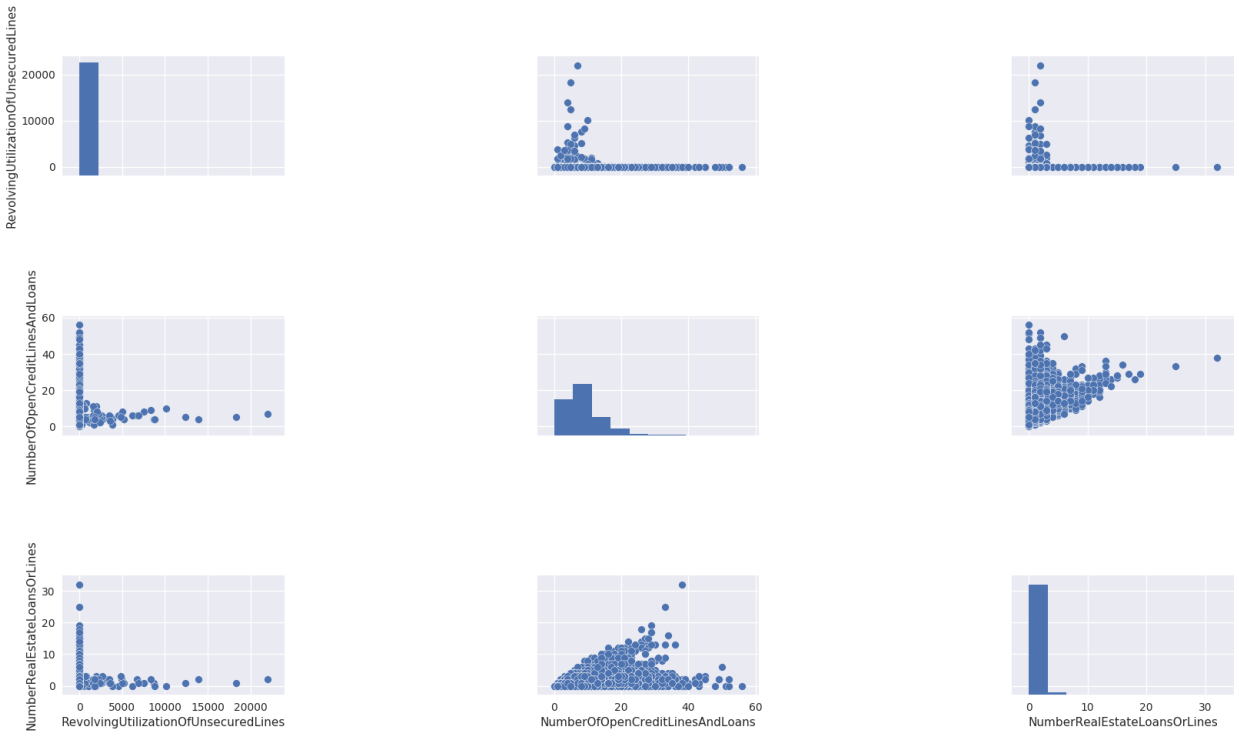
- *varDist(dataFrame,var)*: This function takes in a *dataFrame* object and *var*, the variable you would like to see a distribution plot for and returns a distribution plot of *var*. Running this function on variable *numberOfDependents* results in the plot below:



- *corrSummary(dataFrame)*: This function takes in a *dataFrame* and returns a heatmap of the correlations between all the variables. Running this on the credit data results in the heatmap below.

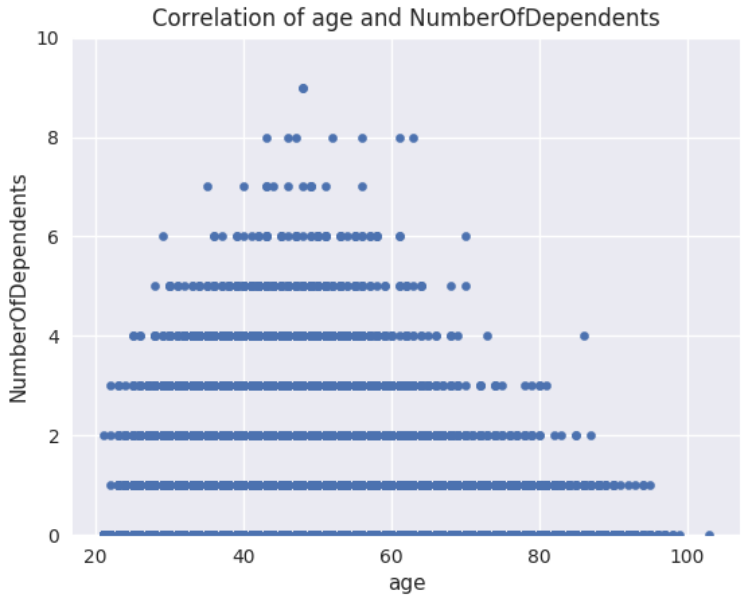


- *plotCorr(dataFrame,list)*: This function takes in a *dataFrame* object and a list of variables you would like to see a correlation for. It returns a plot of all the correlations for the variables provided. Running this on the variables *RevolvingUtilizationOfUnsecuredLines, NumberOfOpenCreditLinesAndLoans, NumberRealEstateLoansOrLines* results in the plot below:



- *findLowOutliers(dataFrame,var, num)*: This function takes in a *dataFrame* object, the variable *var* you want to find low outliers for, and the number of outliers you want *num*. Running this function on *debtRatio* to find the 3 lowest outliers yields the following array: $[-0.25573621, -0.25573621, -0.25573621]$
- *findHighOutliers(dataFrame,var,num)*: This function takes in a *dataFrame* object, the variable *var* you want to find high outliers for, and the number of outliers you want *num*. Running this function on *debtRatio* to find the 10 highest outliers yields the following array: $[18.64412907, 18.71742623, 20.32533443, 23.11834196, 25.08424891,$ $29.6749657, 30.88552614, 37.63658024, 46.20077458, 82.21128292]]$

- *findBivariateOutliers(dataFrame,varX, varY,maxY)*: This function takes in a *dataFrame* object, the variable *varX* you want to be plotted on the x-axis,*varY* you want to be plotted on the y-axis, and *maxY*, the upper limit of *varY*. Running this function *age* and *numberOfDependents* results in the plot below:



3. **Pre-Process Data:**

- *fillMissing(dataFrame,var,newVal)*: This function takes in the variable you want to fill in with missing data *var* and the value you want the missing data to have *newVal*. In *pipeTest.py*, I filled in the missing data for my continuous variables with that variable's median, and for the categorical variable *zipcode*, I filled missing values with 0.

4. **Generate Features and Predictors:**

- *discretize(dataFrame,var,num)*: This function takes in the variable *var* you want to discretize and *num*, the number of bins you want to discretize *var* into. Running this function discretizes *age* into the following 5 bins:
  (a) (20.912, 38.6]
  (b) (38.6, 56.2]
  (c) (56.2, 73.8]
  (d) (73.8, 91.4]
  (e) (91.4, 109.0]
- *createDummy(dataFrame,var)*: This function takes in the variable *var* you want to create dummy variables for. Running this function generates the following dummy variables for *zipcode* [60601, 60618, 60625, 60629, 60637, 60644]

5. **Build Classifier:**

   - *logReg(dataFrame,IV,listOfDVs)*: This function takes in the independent variable *IV* and the dependent variables *listOfDVs* you want to generate a logistic regression for. It returns the coefficients of the logistic regression, as well as its accuracy with the original data. Running this on the credit data resulted in 83.4% accuracy with the original data, as well as the following coefficients:

Intercept	[1.7464243465145182e-06]
RevolvingUtilizationOfUnsecuredLines	[-0.0001633754719177896 8]
age	[-0.03262736739776818]
zipcode	[5.897286708290562e-07]
NumberOfTime30-59DaysPastDueNotWorse	[0.04751821911375464]
DebtRatio	[-7.298373065337037e-05]
MonthlyIncome	[-3.17536592389569e-05]
NumberOfOpenCreditLinesAndLoans	[0.00987689310445205]
NumberOfTimes90DaysLate	[0.03450484254010904]
NumberRealEstateLoansOrLines	[0.006558732565761694]
NumberOfTime60-89DaysPastDueNotWorse")	[0.018844760061179952]
NumberOfDependents	[0.011227895523920409]

6. **Evaluate Classifier:**
   If we evaluate this classifier on accuracy with the original data, then an 83.4% accuracy is reasonably good. At the very least, it does better than random, which results in a 16.7% accuracy. However, since we are only evaluating this model on the same data it trained on, this could be a case of overfitting. We do not know how this model will behave with new data points.