**Predictive Models  
Assignment 1**

All of Assignment 1 should be done in Python and a Juypter notebook.

**Notebook 1: Warm up on Munging and Pre-procesing using Pandas, Matplotlib (and a little Sklearn).**

Consider the data collected for a set of cars. This data is available in .csv format in the file auto.csv . Use **Python**’s **Pandas** (as well as other libraries as mentioned) to perform the following tasks. *You can look at the video store data and code as an example (it is very similar)..* You can check the Pandas documentation as well as many on-line blogs for assistance. http://pandas.pydata.org/pandas-docs/stable/

1. Read in auto.csv file into a dataframe, and call it **Auto**.
2. Replace any missing values with the median value of the variable (feature).
3. Explore the general characteristics of the data, by computing the means and standard deviations of the numerical attributes, as well as the distributions of “drive type” and “fuel type” (both are categorical). You can do this for each individual variable separately or a function to compute for the entire data frame (or both).
4. Use **z-score normalization** to standardize the values of the **Weight** attribute. Show the results side-by-side with the original Weight attribute. [Do not change the original Weight attribute in the table.] Alternatively you may choose to use a [0,1] Min-Max Normalization.You can create this by a manual calculation or use the sklearn.preprocessing package. [http://scikit-learn.org/stable/modules/classes.html - module-sklearn.preprocessing](http://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing)
5. Convert the categorical variables into dummies. Note that this requires converting each categorical attribute into multiple attributes (dummies) and assigning binary values corresponding to the presence or not presence of the attribute value in the original record.
6. Perform basic correlation analysis a correlation matrix) among the attributes. Perform both numeric and visual correlation outputs.
7. Using the new converted dataset- run a PCA. How much variation is captured in 3 Principle Components? (May I suggest you use this function   
   from sklearn.decomposition import PCA
8. Using Matplotlib library (or seaborn or both), create a scatter plot of the (non-normalized) Weight attribute relative to MPG. Be sure that your plot contains appropriate labels for the axes.
9. Using the **hist** function in Matplotlib (or Seaborn or both), create histograms for (non-normalized) Luggage (using 6 bins) and MPG (using 7 bins).
10. Perform a cross-tabulation of the two "fuel type" variables versus the two "Drive type" variables. This requires the aggregation of the occurrences of each genre separately for each gender. You can use whatever appropriate data structure you which to store the results, but you can display it as as a 2 x 2 table with entries representing the counts. Then, use **Matplotlib** to create a bar chart graph to visualize of the relationships between these sets of variables (comparing Regular and Premium customer across the two drive types). Your chart should contain appropriate labels for axes.
11. Create a new dataframe where fuel is “regular” AND MPG is over 21. Call it Auto\_sub. Label each section using markup to state what you are doing and what you see.

Your output for the above should look nice and professional AND well annotated. No pages and pages of junk output. Use heads, tails, and contain output so it looks good. A large part of the grade for this section is professionalism in a Juypter notebook.

For this part, you need a .html file of your code and output.

**Notebook 2- Practice with Scikit Learn**

**You can refer to my example and the sklearn documentation:** [**http://scikit-learn.org/stable/index.html**](http://scikit-learn.org/stable/index.html)**. In fact, feel free to replicate my sklearn code in the Week 1, part 1 folder.**

We are going to perform a basic classification process like you did in ADM, but do it with sklearn.

1. Read in the **Churn Calls dataset**. (500 obs, 2 class target). Call the dataframe **Churn**.
2. Set the target variable as Churn (a yes/no variable).
3. Perform some EDA so you get a feel for the data. Make comments on what you see and how it might affect your analysis.
4. Transform the data so factors are dummied and missing values (if any) are fixed.
5. Create a train and test sample of 70/30.
6. Perform a basic decision tree. Alter at least 2 default arguments. Show your classification and confusion matrix.
7. Cross validate your decision tree. CV=5 or 10.
8. Perform a KNN=3. Show your classification and confusion matrix.
9. Cross Validate your KNN. Comment on what this tells you.
10. Perform a Naïve Bayes. Show your classification and confusion matrix.
11. Cross validate your NB. Comment on what this tells you.
12. Optional- add a Random Forest.

Discuss your best performing algorithm. Also each section should have some thought out comments (done in markup) on what was done and what was found. Look at several of the examples I gave.

Your output for the above should look nice and professional AND well annotated. No pages and pages of junk output. Use heads, tails, and constrain output so it looks good. A large part of the grade for this section is professionalism in a Juypter notebook.

**Notebook 3- GLM modeling and Regularization.**[**http://scikit-learn.org/stable/modules/linear\_model.html**](http://scikit-learn.org/stable/modules/linear_model.html)

Using the dataset in UCO called Crimes and Communities (<https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime> - it has 128 variables and 1994 rows. Read this file in DIRECTLY from the URL address. Cal is Crime. You will create in Scikit learn at least 4 of these GLMs:

-Ridge Regression

-LASSO Regression

-ElecticNet Regression

-Use one other linear model found in the sci-kit learn documentation.

a) Build at least four models from the above list. Make sure your coefficients and the MSE are printed. Also, add in two other regression error metric to help determine the best(better) model. <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics>

b) In your models that has a penalty parameter, tune it with a grid search. Explain what the grid search is doing (I know it is searching for different alphas (the penalty)- so explain why you would do this). Make some explanation of how what occurred as you changed the penalty.

c) On another model, add a cross validation to it. Explain what this is doing.

d) Discuss your optimal model with the optional parameters.

e) optional- get creative and see what else you can do with GLM in Sklearn.

On the above- don’t fret if you get bad results on one or more models. The objective is to get your playing with Scikit and thinking about advanced GLM models used in data mining. Comment (in markup) about how each model performed and what you learned from each penalized (regularized) model.

Enter your answers as brief comments in a markdown code block. Ie everything should be nicely done in one Juyter notebook.

You should create one .html python file for this warm up.

**Notebook 4:**

**Start to think about automating your code**. You should be able to use 98% of the above code and run a second data set through it. Hence I would take your Notebook 3 and “make a copy”. Start with that. Also, think about starting a code notebook.

For Notebook 4, repeat all the steps in Notebook 3 using the Prostate data set. The target is LPSA (log PSA test results). More info on the variables can be found here.

<http://www.inside-r.org/packages/cran/ncvreg/docs/prostate>

You will find a prostate.csv file in the BB folder.

**What to turn into BB as a .zip:**

Notebooks 1-4. All well labeled and documented in markup. 33-50% of all points will come from format and documentation. I want it to look professional and be a good piece on what you explored and found. This is extremely important on notebooks 2-4 (which are all predictive model notebooks).