# HW0

**Problem 1: Processing Tabular Data from File**

In this problem, we practice reading csv formatted data and doing some very simple data exploration.

**Part (a): Reading CSV Data with Numpy**

Open the file 𝚍𝚊𝚝𝚊𝚜𝚎𝚝\_𝙷𝚆𝟶.𝚝𝚡𝚝, containing birth biometrics as well as maternal data for a number of U.S. births, and inspect the csv formatting of the data. Load the data, without the column headers, into an numpy array.

Do some preliminary explorations of the data by printing out the dimensions as well as the first three rows of the array. Finally, for each column, print out the range of the values.

**Prettify your output**, add in some text and formatting to make sure your outputs are readable (e.g. "36x4" is less readable than "array dimensions: 36x4").

**Part (b): Simple Data Statistics**

Compute the mean birth weight and mean femur length for the entire dataset. Now, we want to split the birth data into three groups based on the mother's age:

1. Group I: ages 0-17
2. Group II: ages 18-34
3. Group III: ages 35-50

For each maternal age group, compute the mean birth weight and mean femure length.

**Prettify your output.**

Compare the group means with each other and with the overall mean, what can you conclude?

**Part (c): Simple Data Visualization**

Visualize the data using a 3-D scatter plot. How does your visual analysis compare with the stats you've computed in Part (b)?

In [ ]:



​

**Part (d): Simple Data Visualization (Continued)**

Visualize two data attributes at a time,

1. maternal age against birth weight
2. maternal age against femur length
3. birth weight against femur length

using 2-D scatter plots.

Compare your visual analysis with your analysis from Part (b) and (c).

**Part (e): More Data Visualization**

Finally, we want to visualize the data by maternal age group. Plot the data again using a 3-D scatter plot, this time, color the points in the plot according to the age group of the mother (e.g. use red, blue, green to represent group I, II and III respectively).

Compare your visual analysis with your analysis from Part (a) - (c).

**Problem 2: Processing Web Data**

In this problem we practice some basic web-scrapping using Beautiful Soup.

**Part (a): Opening and Reading Webpages**

Open and load the page (Kafka's The Metamorphosis) at

𝚑𝚝𝚝𝚙://𝚠𝚠𝚠.𝚐𝚞𝚝𝚎𝚗𝚋𝚎𝚛𝚐.𝚘𝚛𝚐/𝚏𝚒𝚕𝚎𝚜/𝟻𝟸𝟶𝟶/𝟻𝟸𝟶𝟶⎯𝚑/𝟻𝟸𝟶𝟶⎯𝚑.𝚑𝚝𝚖

into a BeautifulSoup object.

The object we obtain is a parse tree (a data structure representing all tags and relationship between tags) of the html file. To concretely visualize this object, print out the first 1000 characters of a representation of the parse tree using the 𝚙𝚛𝚎𝚝𝚝𝚒𝚏𝚢() function.

**Part (b): Exploring the Parsed HTML**

Explore the nested data structure you obtain in Part (a) by printing out the following:

1. the content of the head tag
2. the text of the head tag
3. each child of the head tag
4. the text of the title tag
5. the text of the preamble (pre) tag
6. the text of the first paragraph (p) tag

**Part (c): Extracting Text**

Now we want to extract the text of The Metamorphosis and do some simple analysis. Beautiful Soup provides a way to extract all text from a webpage via the 𝚐𝚎𝚝\_𝚝𝚎𝚡𝚝() function.

Print the first and last 5000 characters of the text returned by 𝚐𝚎𝚝\_𝚝𝚎𝚡𝚝(). Is this the content of the novela? Where is the content of The Metamorphosis stored in the BeautifulSoup object?

**Part (d): Extracting Text (Continued)**

Using the 𝚏𝚒𝚗𝚍\_𝚊𝚕𝚕() function, extract the text of all 𝚙 tags and concatenate the result into a single string. Print out the first 1000 characters of the string as a sanity check.

**Part (e): Sentence and Word Count**

Count the number of words in The Metamorphosis. Compute the average word length and plot a histogram of word lengths.

Count the number of sentences in The Metamorphosis. Compute the average sentence length and plot a histogram of sentence lengths.

**Hint**: You'll need to pre-process the text in order to obtain the correct word/sentence length and count.

**Problem 3: Data from Simulations**

In this problem we practice generating data by setting up a simulation of a simple phenomenon, a queue.

Suppose we're interested in simulating a queue that forms in front of a small Bank of America branch with one teller, where the customers arrive one at a time.

We want to study the queue length and customer waiting time.

**Part (a): Simulating Arrival and Service Time**

Assume that gaps between consecutive arrivals are uniformly distributed over the interval of 1 to 20 minutes (i.e. any two times between 1 minute and 6 minutes are equally likely).

Assume that the service times are uniform over the interval of 5 to 15 minutes.

Generate the arrival and service times for 100 customers, using the 𝚞𝚗𝚒𝚏𝚘𝚛𝚖() function from the 𝚛𝚊𝚗𝚍𝚘𝚖 library.

**Part (b): Simulating the Queue**

Write function that computes the average queue length and the average customer wait time, given the arrival times and the service times.

**Part (c): Average Queue Length and Wait Time**

Run your simulation 500 times and report the mean and std of the average wait time and queue length for 100 customers. What do these statistics mean?

Explain why is isn't sufficient to run our simulation **once** and report the average wait time/queue length we obtain.

**Problem 4 (Challenge Problem): More Web Scrapping**

In this problem we practice extracting tabular web data. Open and read the webpage at

𝚑𝚝𝚝𝚙://𝚠𝚠𝚠.𝚝𝚑𝚒𝚜𝚒𝚜𝚖𝚘𝚗𝚎𝚢.𝚌𝚘.𝚞𝚔/𝚖𝚘𝚗𝚎𝚢/𝚗𝚎𝚠𝚜/𝚊𝚛𝚝𝚒𝚌𝚕𝚎⎯𝟸𝟿𝟸𝟾𝟸𝟾𝟻/𝙴𝚌𝚘𝚗𝚘𝚖𝚢⎯𝚝𝚊𝚋𝚕𝚎𝚜⎯𝙶𝙳𝙿⎯𝚛𝚊𝚝𝚎𝚜⎯𝚒𝚗𝚏𝚕𝚊𝚝𝚒𝚘𝚗⎯𝚑𝚒𝚜𝚝𝚘𝚛𝚢⎯𝚞𝚗𝚎𝚖𝚙𝚕𝚘𝚢𝚖𝚎𝚗𝚝.𝚑𝚝𝚖𝚕

Extract the Inflation History table and load it into a numpy array.

Generate a line graph representing the trend of consumer price index vs time (in months).

# HW1

## Problem 1: Basic Data Visualization, Manipulation and Analysis

In this problem, we will be using some basic tools of data visualization and statistical analysis to help build our intuition for finding, describing and interpreting patterns in data. We will be working with the data contained in the file dataset\_HW1.txt. The data in dataset\_HW1.txt contains biometric readings for a number of diabetic patients. In this problem, we will try to detect disease subtypes within this set of patients using their biometric readings.

### Part (a): Understanding a Pandas Dataframe

Load the data from dataset\_HW1.txt into a pandas dataframe. Get a basic picture of the information contained in the dataframe by printing the head, the size of the dataframe, the range of values in numeric-valued columns. Store the max, min and range information as a pandas dataframe.

**Solution:**

### Part (b): Descriptive Statistics

Find the mean, median and standard deviation of the real-valued columns in the entire dataset. Now find the same set of descriptive statistics for each of the following subset of data:

1. children (ages 3 - 17)
2. adult males (ages 18 - 90)
3. adult females (ages 18 - 90)

Append these stats to the dataframe from part (a), containing the max, min and range. Explain what these stats say about the entire dataset as well as each of the above subsets.

Give a summary of the demographics of the patients contained in the dataset, i.e. how many children, how many adults, how many adult females and how many adult males. Display these results as pie charts.

**Solution:**

### Part (c): Data Visualization

To understand how the biometric reading might help us determine diabetic subtypes, we begin with some visual analysis of the biometric data:

1. plot histograms of marker 1 and marker 2 (play with different values for the bin number to find the most usefule visualization)
2. plot both markers in a scatter plot, color each point according to disease subtype

Do the above for each demographics within the data: children, adult men and adult women.

**Solution:**

### Part (d): Putting it All Together

Summarize and interpret the patterns in the **histograms** from Part (c), compare these patterns to each other. In particular, explain what these patterns indicate about the relationship between the biometric data (marker 1 and marker 2) and subtypes of diabetes amongst the patients.

Summarize and interpret the patterns in the **scatter plots** from Part (c), compare these patterns to each other. In particular, explain what these patterns indicate about the relationship between the biometric readings (marker 1 and marker 2) and subtypes of diabetes amongst the patients.

Do your analyses of the patterns in histograms and the scatter plots support or contradict each other? Is one of the two visualizations more useful than the other for anlyzing the relationship between biometric data and disease subtypes? If so which, and why?

Compare the descriptive stats you computed in part (b) with the visualizations. Which aspects of the visualizations (histogram and scatter plots) does each statistic measure?

Do the stats support, contradict or enhance your visual analysis of the biometric data?

Finally, what can you conclude, based on your visual analysis and stats, about the relationship between biometric data and diabetic subtypes in this pool of patients?

**Solution:**

## Problem 2: Introduction to Classification

In this problem, we will use your analysis of the relationship between biometric data and diabetic subtypes from Problem 1 to **classify** the disease subtype of new patients.

### Part (a): A Disease Subtype Classifier for Children

In the pool of child patients, compute the mean biometric data (marker 1 and marker 2) for each diabetic subtype. For a new patient between the ages of 3 and 18, we will classify the patient as the subtype whose biometric mean is most ``similar" to the patient's biometric data.

* Explain why this way of classifying disease subtypes of new patients is reasonable. Support your explanation with your analysis from Problem 1.

To evaluate our classifier, we can use it to classify a set of new patients whose disease subtype we already know. We measure the quality of our classifier by compute the percentage of new patients whose disease subtype we correctly classify.

* Explain why the percentage of correct classification is a reasonable way to evaluate our classifier. What might be some shortcomings or ambiguities of this method of evaluation (**Hint:** think about the effect of the number of patients in each disease subtype, confidence level etc).

**Solution:**

### Part (b): Implementation

* **Randomly** split the child patient data into two sets: training (70%) and testing (30%).
* Implement the scheme for classifying disease subtypes described in Part (a). That is, write a function, classify, that takes as input the training data and testing data (representing new patients), which then:
  1. computes the biometric means for subtypes using the training data
  2. classifies the disease subtype of each new patient by comparing their biometric data to the means you've computed (i.e. compute the Euclidean distance between a new patient's biometric data and the biometric means of each disease subtype, classify the new patient as the subtype whose means is most similar in term of the Euclidean distance)
* Write a function, evaluate, that takes as input the actual disease subtypes for a set of patients as well as the predicted disease subtypes and computes the percentage of new patients who are correctly classified.
* Use the classify function to classify the disease subtypes of your testing data. Then use the evaluate function to evaluate your classification.

**Solution:**

### Part (c): Further Evaluation of Our Classifier

* Create training and testing sets from the adult female patients. Use the classify function from Part (b) to classify the disease subtypes of your testing data. Then use the evaluate function to evaluate your classification.
* Do the same for adult male patients.
* Is our method of classifying disease subtypes valid for adult male and adult female patients (use the returned values of the evaluate function to support your explanation)?
* Explain why our method of classifying disease subtypes is or is not valid for adult data.

**Solution:**

## Problem 3: More Classification

In this problem, we will explore a different way to classify the disease subtype of new patients.

### Part (a): Another Classifier

In Problem 2, we classified the disease subtype of a new patient by comparing their biometric data with the biometric means of the subtypes we compute from the training data. This time, we compare the biometric data of the new patient with the data of all the patients in the training data. We identify the patient in the training data whose biometric data is most similar (in terms of Euclidean distance) to that of the new patient. Finally, we classify the disease subtype of the new patient as that of the patient most similar to them.

Explain why this way of classifying disease subtypes of new patients is reasonable. Support your explanation with your analysis from Problem 1.

**Solution:**

### Part (b): Implementation

* **Randomly** split the child patient data into two sets: training (70%) and testing (30%).
* Implement the scheme for classifying disease subtypes described in Part (a). That is, write a function, classify, that takes as input the training data and data for new patients, which then:
  1. for each new patient, identify the patient in the training set whose biometric data is the most similar to this patient
  2. classifies the disease subtype of each new patient as the subtype of the patient most similar to them
* Use the classify function to classify the disease subtypes of your testing data. Then use the evaluate function, from Problem 2, to evaluate your classification.
* Do the same for adult male and adult female data.

**Solution:**

### Part (c): Comparison of Classifiers

Compare the performance of the classifier you implemented in Part (b) with the one from Problem 2.

Which classifier does a better job on the child patient data? Explain why, using your analysis of the dataset from Problem 1.

Answer the above question for the adult male and adult female data.

**Solution:**

## Challenge Problem: US Voting Data (by Age and Sex)

**(Required for AC 209A Students)**

In this problem, you will perform preliminary data exploration and visualization of some real voting data.

**Note:** You are now working with real-life data, so be cautious regarding data type, data format and data quality.

### Part (a): Downloading and Understanding the Data

Download Table 1 from the [US Census Bereau](http://www.census.gov/data/tables/time-series/demo/voting-and-registration/p20-577.html). This is an excel file (do not perform any analysis using Microsoft Excel).

Load the data into a Python data structure.

Write a brief summary describing what information is included in this dataset (provide evidence to support your summary).

**Solution:**

### Part (b): What's the Story?

Perform data exploration and identify as well as describe the major trends in this data set. Use your computations and visualizations effecitively and specifically to support your analysis, hypothesis and conclusions. Your analysis must include relationships between age, sex and reported percentage of voter registration, reported voting.

**Solution:**

### Part (c): What's Next?

Based on your analysis in part (c), what types of interesting questions or tasks could you ask of or perform with this data (explain your answer)? What additional data do you anticipate needing in order to answer these questions/perform these tasks (explain your answer)?

**Solution:**

# HW2

## Problem 1: Inside the Models in Scikit-learn

In this problem, we will be implementing K-Nearest Neighbour and simple linear regression for predicting a quantitative variable. We will compare the performance of our implementation with those of Scikit-learn (sklearn).

The datasets required for this problem is in the dataset directory. Each file in the dataset directory contains a one-dimensional data set, with the first column containing the dependent variable Y, and the second column containing the independent variable X.

### Part (a): Implement the models by hand

In this part you **may not** use sklearn for any task.

In the following, you may use numpy arrays instead of pandas dataframes.

* Implement a funtion split, which satifies:
  + input: an nx2 dataframe data, a float m
  + return: an nx2 dataframe train and an nx2 dataframe test, consisting of m percent and 100 - m percent of the data, respectively.
* Implement K-Nearest Neighbour for predicting a quantitative variable. That is, write a function, knn\_predict, that satisfies:
  + input: an integer k, an n x 2 dataframe training set train, an n x 1 dataframe testing set test
  + return: an nx2 dataframe, whose first column is that of test and whose second column is the predicted values.
* Implement linear regression for predicting a quantitative variable. That is, write a function linear\_reg\_fit that satisfies:
  + input: an nx2 dataframe training set train
  + return: the coefficients of the linear regression model - a float slope and a float intercept.
* Write a function linear\_reg\_predict that satisfies:
  + input: an nx1 dataframe testing set test, as well as the coefficients of the linear regression model
  + return: an nx2 dataframe, whose first column is that of test and whose second column is the predicted values.
* Implement a function score that satisfies:
  + input: an nx2 dataframe predicted, an nx2 dataframe actual
  + return: R^2 coefficient of the fit of the predicted values.

**Solution:**

### Part (b): Compare with sklearn

* Load the contents of dataset\_1\_full.txt into a pandas dataframe, or numpy array.
* Use your functions from Part (a) to split the data into training and testing sets (70-30). Evaluate how KNN and linear regression each perform on this dataset.
* Use sklearn to split the data into training and testing sets (70-30). Use sklearn to evaluate how KNN and linear regression each perform on this dataset.
* Use Python's time library to measure how well your implementations compare with that of sklearn. What can you do (algorithmically or codewise) to make your implementation faster or more efficient?

**Solution:**

## Problem 2: Handling Missing Data

In this problem, we will be handling the problem of datasets with missing values. Clearly, we cannot simply remove entire rows or columns that contain missing values. In this problem, we explore two different ways to fill in missing values.

The datasets required for this problem is in the dataset directory. Each file in the dataset directory contains a one-dimensional data set, with the first column containing the dependent variable Y, and the second column containing the independent variable X.

In this problem, you **may not** use sklearn or build-in pandas functions to **directly fill in missing values**. Usage of these libraries/pakcages for related tasks is fine.

### Part (a): Model Based Data Imputation

* Describe in detail how predictive models for data (like KNN and simple linear regression) can be used to fill in missing values in a data set.
* Implement your scheme. That is, write code (preferably a function fill or two functions fill\_knn, fill\_lin\_reg), which takes an n x 2dataframe or array with values missing in the 2nd column and fills in these values using KNN and linear regression.
* You need to, also, write code to evaluate the quality of the values you've filled in.

**Solution:**

### Part (b): Which Model is Better?

* For datasets dataset\_1\_missing.txt to dataset\_6\_missing.txt, compare the result of filling in the missing values using KNN and linear regression, using both the R^2 coefficient as well as data visualization.
* Use your analysis to form conjectures regarding the conditions under which KNN performs better than linear regression, under which linear regression performs better than KNN and under which both perform equally (well or poorly). Explain in detail exactly what might cause each model to fail or perform well.
* Using dataset\_1\_missing.txt, explain the impact of the choice of k on the performance of KNN.

Use numerical analysis and data visualization to support every part of your argument.

**Solution:**

## Problem 3: Is the Best (Linear Model) Good Enough?

In this problem, we will specifically look at conditions under which linear regression excels or fails.

The datasets required for this problem is in the dataset directory. Each file in the dataset directory contains a one-dimensional data set, with the first column containing the dependent variable Y, and the second column containing the independent variable X.

### Part (a): Introduction to Residual Plots

* Read dataset\_1\_full.txt. Visualize the dataset and make some initial observations.
* For this data set, what can you say about the following linear fits:
  1. slope = 0.4, intercept = 0.2
  2. slope = 0.4, intercept = 4
  3. linear regression model
* In each case, visualize the fit, compute the residuals, and make a residual plot of predicted values against residuals as well as a residual histogram. What do these plots reveal?
* Calculate the R^2 coefficient for all three fits. What do the erors reveal? How do they compare to the residual plots?

**Solution:**

### Part (b): What do Residual Plots Reveal?

* Read datasets dataset\_2\_full.txt through dataset\_6\_full.txt. In each case, visualize the fit of the linear regression model, compute the residuals, and make a residual plot of predicted values against residuals as well as a residual histogram. What do these plots reveal about the fit of the model?
* Calculate the R^2 coefficient each fit. What do the erors reveal? How do they compare to the residual plots?
* Based on your analysis, form conjectures regarding the precise relationship between the residual plots and the fit of the linear regression model. Conjecture on the precise conditions under which linear regression model is an appropriate model for a given dataset.

**Solution:**

## Challenge Problem: Combining Random Variables

This problem, we explore the distirbution of random variables that result from combining other random variables.

### Part (a): Adding Two Uniformly Distributed Variables

Consider the independent random variables X∼U(0,1) and Y∼U(0,1). Let Z be the random variable Z=X+Y.

What is the distribution of Z (give the pdf for Z)? You should fully explain and support your conlusion.

**Hint:** your solution can be a combination of experimentation, empirical evidence and/or algebra

**Solution:**

### Part (b): Adding Multiple Uniformly Distributed Variables

Consider three independent random variables X1,X2,X3∼U(0,1). Let Z be the random variable Z=X1+X2+X3.

What is the distribution of Z? What if you add 10 or 12 independent (standard) uniformly distributed variables? Conjecture on the distribution of

Z=limn→∞∑i=1nXi

where {Xi} are independent (standard) uniformly distributed variables.

**Hint:** your solution can be a combination of experimentation, empirical evidence and/or algebra

**Solution:**

### Part (c): Combining Normally Distributed Variables

Consider the independent random variables X∼(0,1) and Y∼(0,1). Let Z be the random variable Z=X+Y.

What is the distribution of Z (give the pdf for Z)? You should fully explain and support your conlusion.

**Hint:** use properties of expected value and some experimentation.

**Solution:**

### Part (d): Product of Normally Distributed Variables

Is the product of two normally distributed variables a normally distributed variable? You should fully explain and support your conlusion.

**Solution:**

# HW3

## Problem 1: Multiple linear regression

### Part (a): Implement multiple linear regression from scratch

You are provided a data set containing attributes related to automobiles as well as their corresponding prices. The task is to build a linear regression model from scratch that can estimate the price of an automobile (response variable) using its attributes (predictor variables).

The file dataset\_1\_train.txt contains the training set that you can use to fit a regression model, and the file dataset\_1\_test.txt contains the test set that you can use to evaluate the model. In each file, the first two columns contain the predictors of the automobile, namely 'horsepower' and 'highway MPG', and the last column contains the automobile prices.

* Implement the following two functions from scratch.
  + multiple\_linear\_regression\_fit:
    - takes as input: the training set, x\_train, y\_train
    - fits a multiple linear regression model
    - returns the model parameters (coefficients on the predictors, as an array, and the intercept, as a float).
  + multiple\_linear\_regression\_score:
    - takes model parameters (coefficients and intercept) and the test set, x\_test y\_test, as inputs
    - returns the R^2 score for the model on the test set, along with the predicted y-values.
* Use your functions to predict automobile prices and evaluate your predictions.

**Note:** You **may not** use pre-built models or model evaluators for these tasks.

### Part (b): Confidence interval on regression parameters

Using your linear regression implementation from Part (a), model the data in dataset\_2.txt, which contains five predictor variables in the first five columns, and the response variable in the last column.

Compute confidence intervals for the model parameters you obtain:

* Create 200 random subsamples of the data set of size 100, and use your function to fit a multiple linear regression model to each subsample.
* For each coefficient on the predictor variables: plot a histogram of the values obtained across the subsamples, and calculate the confidence interval for the coefficients at a confidence level of 95%.
* Highlight the mean coeffcient values and the end points of the confidence intervals using vertical lines on the histogram plot. How large is the spread of the coefficient values in the histograms, and how tight are the confidence intervals?
* Use the formula for computing confidence intervals provided in class (or use statmodels) to compute the the confidence intervals. Compare confidence intervals you find through simulation to the ones given by the formula (or statmodels), are your results what you would expect?

**Note:** You **may not** use pre-built models or model evaluators for these tasks.

## Problem 2: Polynomial regression

In this problem, we revisit a dataset from Homework 1 and fit polynomial regression models to it. The dataset is provided in the file dataset\_3.txt, which contains a single predictor variable x in the first column and the response variable y in the second column.

### Part(a): Implement polynomial regression from scratch

* Implement the following three functions from scratch:
  + polynomial\_regression\_fit:
    - takes as input: training set, x\_train, y\_train and the degree of the polynomial
    - fits a polynomial regression model
    - returns the model parameters (array of coefficients and the intercept)
  + polynomial\_regression\_predict:
    - takes as input: the model parameters (array of coefficients and the intercept), the degree of the polynomial and the test set predictors x\_test
    - returns the response values predicted by the model on the test set.
  + polynomial\_regression\_score:
    - takes an array of predicted response values and the array of true response values y\_test
    - returns R^2 score for the model on the test set, as well as the sum of squared errors
* Fit polynomial regression models of degrees 3, 5, 10 and 25 to the data set. Visualize the original data along with the fitted models for the various degrees in the same plot.

For this problem, you may either use the multiple linear regression functions implemented in the Problem 1 or use the in-built functions in sklearn.

### Part (b): Comparing training and test errors

* Split the data set in Problem 2 each into training and test sets: use the first 50% of the data for training and the remaining for testing.
* Fit polynomial models of varying degree ranging from 1 to 15 to the training sets. Evaluate the various fits on **both** the training and the test sets. Plot both the R^2 score of the fitted polynomial models on the training and test sets as a functions of the degree.
* Describe the relationship between degree of the polynomial model and the fit on both the training and testing data. Explain, based on the plot, what is the best polynomial model for the data.

## Problem 3: Model selection criterion

In this problem, we examine various criteria that help us decide how to choose between multiple models for the same data.

### Part (a): How does one choose the best polynomial degree?

In Problem 2, you fitted polynomials of different degrees to the entire data set, and inspected the quality of fits on the test set. In practice, one needs to find the 'best' model for the given prediction task using **only** the training set. For this, we'll now make use of two model selection criteria, namely, the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). These are evaluated on the training set, but serve as a proxy for the test set accuracy.

For dataset\_3.txt, do the following:

* For each polynomial model you fitted, compute the AIC and BIC for the model on the training set. Plot the criterion values as a function of the polynomial degree.
* Which model is chosen by each criterion? Do they match with the model that yields maximum test R^2 score?

### Part (b): Application to New York taxi cab density estimation

We shall now apply the concepts learned so far to a real-world prediction task. You are asked to build a regression model for estimating the density of Green cab taxis at any given time of a day in New York city. The model needs to take the time of the day (in minutes) as input, and predict the expected number of pick ups at that time.

The data set for this problem can be downloaded from the following URL: <https://s3.amazonaws.com/nyc-tlc/trip+data/green_tripdata_2015-01.csv>. The file contains the details of all pickups by Green cabs in New York City during January 2015.

## Challenge Problem: Advanced regression techniques

In this problem, we revisit the automobile pricing data set in Problem 1(a) and explore advanced regression techniques to build better models.

### Part (a): Polynomial regression on multi-dimensions

In Problems 2-3, you had implemented a polynomial regression technique for data sets with a single predictor variable. How would you use a similar approach to fit a polynomial model on data sets with more than one predictor?

Reload dataset\_1\_train.txt and dataset\_1\_test.txt. Fit polynomial models of degrees 2 and 3 to the training set, and evaluate the R^2 score of the fitted model on the test set. How do they compare with the test performance of a linear regression model?

### Part (b): Weighted linear regression

Suppose you are told that some of the prices recorded in the training set are noisy, and you are given the list of noisy points, how would you use this information during training to fit a better regression model?

The noise level for each training point is provided in the file dataset\_1\_train\_noise\_levels.txt. A noise level 'none' indicates that the price is accurate, and a noise level 'noisy' indicates that the price is only moderately accurate.

We want to fit a linear regression model that accounts for this new information. One way to do this is to assign different weights to each training point based on the amount of noise associated to that training point. That is, our loss function is now

∑i=1nαi(yi−wTxi)2

where αi is a number representing how much you value the contribution of the data point xi.

How does the R^2 score (evaluated on the test set) of the new linear model compare to the one fitted using plain linear regression?

# HW4

## Problem 1: Variable selection and regularization

The data set for this problem is provided in dataset\_1.txt and contains 10 predictors and a response variable.

### Part (a): Analyze correlation among predictors

* By visually inspecting the data set, do find that some of the predictors are correlated amongst themselves?
* Compute the cofficient of correlation between each pair of predictors, and visualize the matrix of correlation coefficients using a heat map. Do the predictors fall naturally into groups based on the correlation values?
* If you were asked to select a minimal subset of predictors based on the correlation information in order to build a good regression model, how many predictors will you pick, and which ones will you choose?

### Part (b): Selecting minimal subset of predictors

* Apply the variable selection methods discussed in class to choose a minimal subset of predictors that yield high prediction accuracy:
  + Exhaustive search
  + Step-wise forward selection **or** Step-wise backward selection

    In each method, use the Bayesian Information Criterion (BIC) to choose the subset size.

* Do the chosen subsets match the ones you picked using the correlation matrix you had visualized in Part (a)?

**Note**: You may use the statsmodels's OLS module to fit a linear regression model and evaluate BIC. You may **not** use library functions that implement variable selection.

### Part (c): Apply Lasso and Ridge regression

* Apply Lasso regression with regularization parameter λ=0.01 and fit a regression model.
  + Identify the predictors that are assigned non-zero coefficients. Do these correspond to the correlation matrix in Part (a)?
* Apply Ridge regression with regularization parameter λ=0.01 and fit a regression model.
  + Is there a difference between the model parameters you obtain different and those obtained from Lasso regression? If so, explain why.
  + Identify the predictors that are assigned non-zero coefficients. Do these correspond to the correlation matrix in Part (a)?
* Is there anything peculiar that you observe about the coefficients Ridge regression assigns to the first three predictors? Do you observe the same with Lasso regression? Give an explanation for your observation.

**Note**: You may use the statsmodels or sklearn to perform Lasso and Ridge regression.

## Problem 2: Cross-validation and Bootstrapping

In this problem, you will work with an expanded version of the automobile pricing data set you analyzed in Homework 3. The data set is contained dataset\_2.txt, with 26 attribues (i.e. predictors) for each automobile and corresponding prices.

### Part(a): Encode categorical attributes and fill missing values

Identify the categorical attributes in the data. Replace their values with the one-hot binary encoding. You may do this using the get\_dummies() function in pandas. If you do this task correctly, you should get a total of 69 predictors after the encoding.

### Part (b): Apply regular linear regression

* Split the data set into train and test sets, with the first 25% of the data for training and the remaining for testing.
* Use regular linear regression to fit a model to the training set and evaluate the R^2 score of the fitted model on both the training and test sets. What do you observe about these values?
* You had seen in class that the R^2 value of a least-squares fit to a data set would lie between 0 and 1. Is this true for the test R^2 values reported above? If not, give a reason for why this is the case.
* Is there a need for regularization while fitting a linear model to this data set?

**Note**: You may use the statsmodels or sklearn to fit a linear regression model and evaluate the fits.

### Part (c): Apply Ridge regression

* Apply Ridge regression on the training set for different values of the regularization parameter λ in the range {10−7,10−6,…,107}. Evaluate the R^2 score for the models you obtain on both the train and test sets. Plot both values as a function of λ.
* Explain the relationship between the regularization parameter and the training and test R^2 scores.
* How does the best test R^2 value obtained using Ridge regression compare with that of plain linear regression? Explain.

**Note**: You may use the statsmodels or sklearn to fit a ridge regression model and evaluate the fits.

### Part (d): Tune regularization parameter using cross-validation and bootstrapping

* Evaluate the performance of the Ridge regression for different regularization parameters λ using 5-fold cross validation **or** bootstrapping on the training set.
  + Plot the cross-validation (CV) or bootstrapping R^2 score as a function of λ.
  + How closely does the CV score or bootstrapping score match the R^2 score on the test set? Does the model with lowest CV score or bootstrapping score correspond to the one with maximum R^2 on the test set?
  + Does the model chosen by CV or bootstrapping perform better than plain linear regression?

**Note**: You may use the statsmodels or sklearn to fit a linear regression model and evaluate the fits. You may also use kFold from sklearn.cross\_validation.

## Problem 3: Ridge regression via ordinary least-squares regression

We present an approach to implement Ridge regression using oridinary least-squares regression. Given a matrix of responses X∈ℝn×p and response vector y∈ℝn, one can implement Ridge regression with regularization parameter λ as follows:

* Augment the matrix of predictors X with p new rows containing the scaled identity matrix λ⎯⎯√I∈ℝp×p, i.e.

X⎯⎯⎯⎯=X11⋮Xn1λ⎯⎯√⋮0…⋱……⋱…X1p⋮Xnp0⋮λ⎯⎯√∈ℝ(n+p)×p.

* Augment the response vector y with a column of p zeros, i.e.

y⎯⎯⎯=y1⋮yn0⋮0∈ℝn+p.

* Apply ordinary least-squares regression on the augmented data set (X⎯⎯⎯⎯,y⎯⎯⎯).

### Part (a): Show the proposed approach implements Ridge regression

Show that the approach proposed above implements Ridge regression with parameter λ.

### Part (b): Debug our implementation of ridge regression

You're a grader for CS109A, the following is an implemention of Ridge regression (via the above approach) submitted by a student. The dataset is dataset\_3.txt. The regression model is fitted to a training set, and the R^2 scores of the fitted model on the training and test sets are plotted as a function of the regularization parameter. Grade this solution according to the following rubric (each category is equally weighted):

* correctness
* interpretation (if applicable)
* code/algorithm design
* presentation

In addition to providing an holistic grade (between 0 to 5), provide a corrected version of this code that is submission quality.

*# Fit*

**def** ridge(x\_train, y\_train, reg\_param):

n=np.shape(x\_train)[0]

x\_train=np.concatenate((x\_train,reg\_param**\***np.identity(n)),axis=1)

y\_train\_=np.zeros((n**+**np.shape(x\_train)[1],1))

**for** c **in** range(n):

y\_train\_[c]= y\_train[c]

**import** sklearn

model = sklearn.linear\_model.LinearRegression()

model.fit(x\_train,y\_train.reshape(**-**1,1))

**return** model

​

*# Score*

**def** score(m,x\_test,y\_test, reg\_param):

n=np.shape(x\_train)[0]

x\_test=np.concatenate((x\_test,reg\_param**\***np.identity(n)),axis=1)

y\_test\_=np.zeros((n**+**np.shape(x\_test)[1],1))

**for** c **in** range(n):

y\_test\_[c]= y\_test[c]

**return** m.score(x\_test,y\_test.reshape(**-**1,1))

​

*# Load*

data = np.loadtxt('datasets/dataset\_3.txt', delimiter=',')

n = data.shape[0]

n = int(np.round(n**\***0.5))

x\_train = data[0:n,0:100]

y\_train = data[0:n,100]

x\_test = data[n:2**\***n,0:100]

y\_test = data[n:2**\***n,100]

​

*# Params*

a=np.zeros(5)

**for** i **in** range(**-**2,2):

a[i**+**2]=10**\*\***i

​

*# Iterate*

rstr =np.zeros(5)

rsts =np.zeros(5)

**for** j **in** range(0,5):

m =ridge(x\_train,y\_train,a[i])

rstr[j]=score(m,x\_train,y\_train,a[j])

rsts[i]=score(m,x\_test,y\_test,a[i])

​

*# Plot*

plt.plot(a,rstr)

plt.plot(a,rsts)

Out[91]:

[<matplotlib.lines.Line2D at 0x1149841d0>]

## Challenge Problem: Predicting Outcome of a Fund-raising Campaign

You are provided a data set containing details of mail sent to 95,412 potential donors for a fund-raising campaign of a not-for-profit organization. This data set also contains the amount donated by each donor. The task is to build a model that can estimate the amount that a donor would donate using his/her attributes. The data is contained in the file dataset\_4.txt. Each row contains 376 attributes for a donor, followed by the donation amount.

**Note**: For additional information about the attributes used, please look up the file dataset\_4\_description.txt. This files also contains details of attributes that have been omitted from the data set.

### Part (a): Fit regression model

Build a suitable model to predict the donation amount. How good is your model?

### Part (b): Evaluate the total profit of the fitted model

Suppose you are told that the cost of mailing the donor is $7. Use your model to maximize profit. Implement, explain and rigorously justify your strategy. How does your strategry compare with blanket mailing everyone.

### Part (c): Further Discussion

In hindsight, thoroughly discuss the appropriatenes of using a regression model for this dataset (you must at least address the suitability with respect to profit maximization and model assumptions). Rigorously justify your reasoning.

# HW5

## Problem 1: Image Classification

In this problem, your task is to classify images of handwritten digits.

The data set is provided in the file dataset\_1.txt and contains 8x8 gray-scale images of hand-written digits, flattened to a 64-length vector. The last column contains the digit. For simplicity, we have only included digits 0, 1 and 3.

We want you to build a model that can be given the image of a hand-written digit and correctly classify this digit as 0, 1 or 3.

### Part 1(a). Reduce the data

Images data are typically high dimensional (the image vector has one feature for every pixel). Thus, to make working with image data more tractible, one might first apply a dimension reduction technique to the data.

* Explain why PCA is a better choice for dimension reduction in this problem than step-wise variable selection.
* Choose the smallest possible number of dimensions for PCA that still permits us to perform classification.

(**Hint:** how do we visually verify that subgroups in a dataset are easily classifiable?)

* Visualize and interpret the principal components. Interpret, also, the corresponding PCA varaiable values.

### Part 1(b). Build a classifier

So far, we have only learned models that distinguishes between two classes. Develop and implement a **simple and naive** method of distinguishing between the three digits in our reduced dataset using binary classifiers.

### Part 1(c). Build a better one

Asses the quality of your classifier.

* What is the fit (in terms of accuracy or R^2) of your model on the reduced dataset? Visually assess the quality of your classifier by plotting decision surfaces along with the data. Why is visualization of the decision surfaces useful? What does this visualization tell you that a numberical score (like accuracy or R^2) cannot?
* What are the draw backs of your approach to multi-class classification? What aspects of your method is contributing to these draw backs, i.e. why does it fail when it does?

(**Hint:** make use your analysis in the above; think about what happens when we have to classify 10 classes, 100 classes)

* Describe a possibly better alternative for fitting a multi-class model. Specifically address why you expect the alternative model to outperform your model.

(**Hint:** How does sklearn's Logistic regression module handle multiclass classification?).

## Problem 2. Sentiment Analysis

In this problem, you will explore how to predict the underlying emotional tone of textual data - this task is called sentiment analysis.

You will be using the dataset in the file dataset\_2.txt. In this dataset, there are 1382 posts containing textual opinions about Ford automobiles, along with labels indicating whether the opinion expressed is positive or negative.

Given a new post about an automobile, your goal is to predict if the sentiment expressed in the new post is positive or negative. For this task you should implement a regularized logistic regression model.

Produce a report summarizing your solution to this problem:

* Your report should address all decisions you made in the "Data Science Process" (from Lectures #0, #1, #2):

a. Data collection & cleaning

b. Data exploration

c. Modeling

d. Analysis

e. Visualization and presentation

* Your report should be informative and accessible to a **general audience with no assumed formal training in mathematics, statistics or computer science**.
* The exposition in your report, not including code, visualization and output, should be at least three paragraphs in length (you are free to write more, but you're not required to).

Structure your presentation and exposition like a professional product that can be submitted to a client and or your supervisor at work.

## Challenge Problem: Automated Medical Diagnosis

In this problem, you are going to build a model to diagnose heart disease.

The training set is provided in the file dataset\_3\_train.txt and there are two test sets: dataset\_3\_test\_1.txt and dataset\_3\_test\_2.txt. Each patient in the datasets is described by 5 biomarkers extracted from cardiac SPECT images; the last column in each dataset contains the disease diagnosis (1 indicates that the patient is normal, and 0 indicates that the patient suffers from heart disease).

* Fit a logistic regression model to the training set, and report its accuracy on both the test sets.
* Is your accuracy rate meaningful or reliable? How comfortable would you be in using your predictions to diagnose real living patients? Justify your answers.

(**Hint:** How does the performance of your model compare with a classifier that lumps all patients into the same class?)

* Let's call the logistic regression model you learned, C1. Your colleague suggests that you can get higher accuracies for this task by using a threshold of 0.05 on the Logistic regression model to predict labels instead of the usual threshold of 0.5, i.e. use a classifier that predicts 1 when Pˆ(Y=1|X)≥0.05 and 0 otherwise. Let's call this classifier C2. Does C2 perform better the two test sets - that is, which one would you rather use for automated diagnostics? Support your conclusion with careful analysis.
* Generalize your analysis of these two classifiers. Under what general conditions does C2 perform better than C1? Support your conclusion with a mathematical proof or simulation

**Hint:** You were told in class that a classifier that predicts 1 when Pˆ(Y=1|X)≥0.5, and 0 otherwise, is the Bayes classifier. This classifier minimizes the classification error rate. What can you say about a classifier that uses a threshold other than 0.5? Is it the Bayes classifier for a different loss function?

**Hint:** For the first three parts, you might find it useful to analyze the conditional accuracy on each class.

# HW6

## Problem 1: Recommender System for Movies

In this problem, you will build a model to recommend movies using ratings from users.

The dataset for this problem is contained in dataset\_4\_ratings.txt. This dataset contains ratings from 100 users for 1000 movies. The first two columns contain the user and movie IDs. The last column contains a 1 if the user liked the movie, and 0 otherwise. Not every movie is rated by every user (i.e. some movies have more ratings than others).

The names of the movies corresponding to the IDs are provided in dataset\_4\_movie\_names.txt.

### Part 1(a): Exploring how to rank

One way of recommending movies is to recommend movies that are generally agreed upon to be good. But how do we measure the "goodness" or "likability" of a movie?

* **Implementation:** Suppose we measure the "goodness" of a movie by the probability that it will be liked by a user, P(label=like|movie)=θmovie. Assuming that each user independently rates a given movie according to the probability θmovies. Use a reasonable estimate of θmovies to build a list of top 25 movies that you would recommend to a new user.

**Hint:** What does the likelihood function, P(likes=k|θmovie,n,movie), look like? What θmovie will maximize the likelihood?

* **Analysis:** Why is using θmovie to rank movies more appropriate than using the total number of likes? Explain why your estimate of θmovie is reasonable. Explain the potential draw backs of estimating θmovie this way.

**Hint:** Under what conditions may models that maximize the likelihood be suboptimal? Do those conditions apply here?

### Part 1(b): Exploring the effect of prior beliefs

Let's add a prior, p(θmovie), to our probabilistic model for movie rating. To keep things simple, we will restrict ourselves to using beta priors.

* **Analysis:** How might adding a prior to our model benifit us in our specific task? Why are beta distributions appropriate priors for our application?

**Hint:** Try visualizing beta priors a=b=1, a=b=0.5, a=b=2 and a=4,b=2, for example, what kind of plain-English prior beliefs about the movie does each beta pdf encode?

* **Implementation/Analysis:** How does the choice of prior affect the posterior distribution of the 'likability' for the movies: Toy Story, Star Wars, The Shawshank Redemption, Down Periscope and Chain Reaction.

**Hint:** Use our posterior sampling function to visualize the posterior distribution.

* **Implementation/Analysis:** How does the effect of the prior on the posterior distribution vary with the number of user ratings?

**Hint:** Visualize the posterior distribution for different sizes of subsample of user ratings for the movie Star Wars.

In the following, we've provide you a couple of functions for visualize beta priors and approximating their associated posteriors.

*#-------- plot\_beta\_prior*

*# A function to visualize a beta pdf on a set of axes*

*# Input:*

*# a (parameter controlling shape of beta prior)*

*# b (parameter controlling shape of beta prior)*

*# color (color of beta pdf)*

*# ax (axes on which to plot pdf)*

*# Returns:*

*# ax (axes with plot of beta pdf)*

​

**def** plot\_beta\_prior(a, b, color, ax):

rv = sp.stats.beta(a, b)

x = np.linspace(0, 1, 100)

ax.plot(x, rv.pdf(x), '-', lw=2, color=color, label='a=' **+** str(a) **+** ', b=' **+** str(b))

ax.set\_title('Beta prior with a=' **+** str(a) **+** ', b=' **+** str(b))

ax.legend(loc='best')

**return** ax

*#-------- sample\_posterior*

*# A function that samples points from the posterior over a movie's*

*# likability, given a binomial likelihood function and beta prior*

*# Input:*

*# a (parameter controlling shape of beta prior)*

*# b (parameter controlling shape of beta prior)*

*# likes (the number of likes in likelihood)*

*# ratings (total number of ratings in likelihood)*

*# n\_samples (number of samples to take from posterior)*

*# Returns:*

*# post\_samples (a array of points from the posterior)*

​

**def** sample\_posterior(a, b, likes, ratings, n\_samples):

post\_samples = np.random.beta(a **+** likes, b **+** ratings **-** likes, n\_samples)

**return** post\_samples

### Part 1(c): Recommendation based on ranking

* **Implementation:** Choose a reasonable beta prior, choose a reasonable statistic to compute from the posterior, and then build a list of top 25 movies that you would recommend to a new user based on your chosen posterior statistic.
* **Analysis:** How does your top 25 movies compare with the list you obtained in part(a)? Which method of ranking is better?
* **Analysis:** So far, our estimates of the 'likability' for a movie was based on the ratings provided by all users. What can be the draw back of this method? How can we improve the recommender system for individual users (if you feel up to the challenge, implement your improved system and compare it to the one you built in the above)?

## Problem 2: Predicting Urban Demographic Changes

### Part 2(a): Temporal patterns in urban demographics

In this problem you'll work with some neighborhood demographics of a region in Boston from the years 2000 to 2010.

The data you need are in the files dataset\_1\_year\_2000.txt, ..., dataset\_1\_year\_2010.txt. The first two columns of each dataset contain the adjusted latitude and longitude of some randomly sampled houses. The last column contains economic status of a household:

0: low-income,

1: middle-class,

2: high-income

Due to the migration of people in and out of the city, the distribution of each economic group over this region changes over the years. The city of Boston estimates that in this region there is approximately a 25% yearly increase in high-income households; and a 25% decrease in the remaining population, with the decrease being roughly the same amongst both the middle class and lower income households.

Your task is to build a model for the city of Boston that is capable of predicting the economic status of a household based on its geographical location. Furthermore, your method of prediction must be accurate over time (through 2010 and beyond).

**Hint:** look at data only from 2000, and consider using both Linear Discriminant Analysis (LDA) and Logistic Regression. Is there a reason one method would more suited than the other for this task?

**Hint:** how well do your two models do over the years? Is it possible to make use of the estimated yearly changes in proportions of the three demographic groups to improve the predictive accuracy of each models over the years?

To help you visually interpret and assess the quality of your classifiers, we are providing you a function to visualize a set of data along with the decision boundaries of a classifier.

*#-------- plot\_decision\_boundary*

*# A function that visualizes the data and the decision boundaries*

*# Input:*

*# x (predictors)*

*# y (labels)*

*# poly\_flag (a boolean parameter, fits quadratic model if true, otherwise linear)*

*# title (title for plot)*

*# ax (a set of axes to plot on)*

*# Returns:*

*# ax (axes with data and decision boundaries)*

​

**def** plot\_decision\_boundary(x, y, model, poly\_flag, title, ax):

*# Plot data*

ax.scatter(x[y **==** 1, 0], x[y **==** 1, 1], c='b')

ax.scatter(x[y **==** 0, 0], x[y **==** 0, 1], c='r')

*# Create mesh*

interval = np.arange(0,1,0.01)

n = size(interval)

x1, x2 = meshgrid(interval, interval)

x1 = x1.reshape(**-**1, 1)

x2 = x2.reshape(**-**1, 1)

xx = np.concatenate((x1, x2), axis=1)

​

*# Predict on mesh points*

**if**(poly\_flag):

quad\_features = preprocessing.PolynomialFeatures(degree=2)

xx = quad\_features.fit\_transform(xx)

yy = model.predict(xx)

yy = yy.reshape((n, n))

​

*# Plot decision surface*

x1 = x1.reshape(n, n)

x2 = x2.reshape(n, n)

ax.contourf(x1, x2, yy, alpha=0.1)

*# Label axes, set title*

ax.set\_title(title)

ax.set\_xlabel('Latitude')

ax.set\_ylabel('Longitude')

**return** ax

### Part 2(b): Geographic patterns in urban demographics

In dataset\_2.txt and dataset\_3.txt you have the demographic information for a random sample of houses in two regions in Cambridge. There are only two economic brackets for the households in these datasets:

0: low-income or middle-class,

1 - high-income.

For each region, recommend a classification model, chosen from all the ones you have learned, that is most appropriate for classifying the demographics of households in the region.

**Hint:** Support your answers with both numerical and visual analysis.

## Challenge Problem: Regularization

We have seen ways to include different forms of regularizations in Linear regression and Logistic regression, in order to avoid overfitting. We will now explore ways to incorporate regularization within the discriminant analysis framework.

* When we have a small training sample, we end up with poor estimates of the class proportions πi and covariance matrices Σ. How can we regularize these quantities to improve the quality of the fitted model?
* We have seen that different assumptions on the covariance matrix results in either a linear or quadratic decision boundary. While the former may yield poor prediction accuracy, the latter could lead to over-fitting. Can you think of a suitable way to regularize the covariance to have an intermediate fit?

The solutions that you suggest must include a parameter that allows us to control the amount of regularization.

Be detailed in your explanation and support your reasoning fully. You do not, however, need to implement any of these solutions.

# HW7

## Problem 1: Monitoring Land Cover Changes Using Satellite Images

In the face of rapid urban development and climate change, it is now more urgent than ever for governments (and other organizations) to have a detailed, accurate and up-to-date picture of land use and land cover, as well as how the land use/cover is changing over time, in order to make effective policy decision to manage and protect natural resources. Building such a comprehensive picture of land use/cover for a large region is extremely difficult.

Recent improvements in satellite imagery and image process have allowed for new tools in land use/cover analysis. The following is an image of the change in vegetation cover around Belize from 1975 to 2007:

In this problem, we will explore how to use classifiers to detect the presence and location of vegetation in satellite images.

### Part 1(a): Detecting vegetation in satellite images

The following files contain sampled locations from satelite aeriel images: dataset\_1.txt, ... dataset\_4.txt. The first two columns contain the normalized latitude and longitude values. The last column indicates whether or not the location contains vegetation, with 1 indicating the presence of vegetaion and 0 indicating otherwise.

These small sets of labels are typically generated by hand (that is, locations might be classified based on field studies or by cross-referencing with government databases). Your task is to use the labeled locations to train a model that will predict whether a new location is vegetation or non-vegetation.

* Suppose we were asked to write a computer program to automatically identify the vegetation regions on the landscape. How can we use the model fitting algorithms you have studied so far to identify the boundaries of the vegetation regions? In particular, discuss the suitability of the following algorithms for each of the four data sets (**you do not need to evaluate your classifier, build your argument using data and decision boundary visualizations**):
  + linear or polynomial linear regression
  + linear or polynomial logistic regression
  + linear or quadratic discriminant analysis
  + decision trees
* By a quick visual inspection of each data set, what do you think is the smallest depth decision tree that would provide a good fit of the vegetation boundaries in each case? Does sklearn's decision tree fitting algorithm always provide a good fit for the proposed depth? If not, explain why. **Support your answer with suitable visualization**.

We provide you with a function plot\_tree\_boundary to visualize a decision tree model on the data set.

*#-------- plot\_tree\_boundary*

*# A function that visualizes the data and the decision boundaries*

*# Input:*

*# x (array of predictors)*

*# y (array of labels)*

*# model (the decision tree you want to visualize, already fitted)*

*# title (title for plot)*

*# ax (a set of axes to plot on)*

*# Returns:*

*# ax (axes with data and decision boundaries)*

​

**def** plot\_tree\_boundary(x, y, model, title, ax):

*# PLOT DATA*

ax.scatter(x[y**==**1,0], x[y**==**1,1], c='green')

ax.scatter(x[y**==**0,0], x[y**==**0,1], c='white')

*# CREATE MESH*

interval = np.arange(0,1,0.01)

n = np.size(interval)

x1, x2 = np.meshgrid(interval, interval)

x1 = x1.reshape(**-**1,1)

x2 = x2.reshape(**-**1,1)

xx = np.concatenate((x1, x2), axis=1)

​

*# PREDICT ON MESH POINTS*

yy = model.predict(xx)

yy = yy.reshape((n, n))

​

*# PLOT DECISION SURFACE*

x1 = x1.reshape(n, n)

x2 = x2.reshape(n, n)

ax.contourf(x1, x2, yy, alpha=0.1, cmap='Greens')

*# LABEL AXIS, TITLE*

ax.set\_title(title)

ax.set\_xlabel('Latitude')

ax.set\_ylabel('Longitude')

**return** ax

### Part 1(b). What is the best splitting criterion for decision trees?

Suppose you are given a data set with 100 points in a satellite image, of which 51 are class 1 and 49 are class 0. Consider following two candidate splits for constructing a decision tree:

1. [Part 1 = (Class 1: 11, Class 0: 37), Part 2 = (Class 1: 40, Class 0: 12)]
2. [Part 1 = (Class 1: 25, Class 0: 48), Part 2 (Class 1: 26, Class 0: 1)]

Which of these is a better split according classification error, Gini coefficient, and Entropy criteria? Do the three criteria agree on the best split, or is one better than the other? Support your answer with a concrete explanation.

## Problem 2: Loan Risk Assessment

In this problem, you are asked by an Unamed National Bank to build a risk assessment model that predicts whether or not it is risky to give a loan to an applicant based on the information provided in their application. Traditionally, loan applications are processed and assessed by hand, but now the bank wants to move to an automated loan processing system. That is, the bank will provide you with loan applications that it has processed in the past for you to build a classifier for risk assessment, going forward, the bank will reject the loan applications from applicants labeled risky and approve the applications that are labeled safe by your model.

The relevant training and test sets are provided in the files: dataset\_5\_train.txt and dataset\_5.test.txt. The training and testing sets are created from both approved and rejected loan applications that the bank has processed by hand in the past. The first 24 columns contain attributes for each applicant gathered from their application, and the last column contains the credit risk assessment with 1 indicating that the customer is a loan risk, and 0 indicating that the customer is not a loan risk. The names of the attributes are provided in the file dataset\_5\_description.txt.

### Part 2(a): A simple decision tree model

* Fit a simple decision tree of depth 2 to the training set and report its accuracy on the test set.
* Interpret the way your model performs risk classifcation. Would you recommend this classifier to Unamed National Bank for making decisions on the loan applications of **real people**? If yes, make an argument for the merrits of this classifer. If no, then make necessary changes to the data set and fit a new classifier that you believe is fair to use in practice, then compare the two classifiers.

We have provided you with a function display\_dt to display the structure of the decision tree in DOT format.

*# Print decision tree model 'model', already fitted*

**def** display\_dt(model):

dummy\_io = StringIO.StringIO()

tree.export\_graphviz(dt, out\_file = dummy\_io)

**print** dummy\_io.getvalue()

### Part 2(b): An ensemble of decision trees

* One way to improve the prediciton accuracy for this task is to use an ensemble of decision trees fitted on random samples, as follows: given a training set of size n, sample new training sets uniformly with replacement, and fit a decision tree model on each random sample.

Now, how would you combine the ensemble into a single classifier? There are at lease two ways:

* + Random classifier: predict using a randomly chosen decision tree from the ensemble
  + Majority classifier: predict using the majority vote from decision trees in the ensemble
* We can also fit a Random Forest model for our data (sklearn.ensemble.RandomForestClassifier).

Is there a significant difference in the prediction accuracies of the above three approaches on the loan data set? If so, explain why.

**Note:** The Random Forest approach can easily overfit the training set. What are the important parameters in sklearn's Random Forest fitting function that influence the model fit? For the risk assessment task, you **need** to fit your random forest model by using a suitable model selection procedure to tune these parameters.

## Challenge Problem: Boosting for Classification

We've seen in class that boosting is a useful ensemble method to combine a collection of simple regression trees into a powerful regression model. Chapter 10.1 of the text book ([J.H. Friedman, R. Tibshirani, and T. Hastie, "The Elements of Statistical Learning"](http://statweb.stanford.edu/~tibs/ElemStatLearn/)) describes the boosting technique for classification trees. Implement the method from scratch.

Write a function fit\_and\_score\_boosted\_trees satisfying:

* Input:
  + x\_train: Array of predictors in training set
  + y\_train: Array of binary responses in training set
  + x\_test: Array of predictors in training set
  + y\_test: Array of binary responses in training set
  + M: Number of iterations / Number of decision trees in the ensemble
  + depth: Depth of each decision tree
* Fits an ensemble of T decision trees to the training set
* Output:
  + test\_accuracy: classification accuracy of the ensemble on the test set

Your function will also have to **standardise** the predictors in the training and test sets before applying boosting.

**Hints:**

* sklearn's decision tree learning routine has an option to specific weights on the training points
* sklearn's classifiers make predictions in {0,1} while the book assumes predictions in {-1, 1}

Your implementation will be evaluated based on three test cases:

challenge\_testcase\_1\_train.txt, challenge\_testcase\_1\_test.txt

challenge\_testcase\_2\_train.txt, challenge\_testcase\_2\_test.txt

challenge\_testcase\_3\_train.txt, challenge\_testcase\_3\_test.txt

These cases represent extreme examples of data (each dataset contains a particular type of pathology) that might break an implementaiton that is not carefully thought through.

**Run the code given below to test your implementation. Call test\_implementation and pass it your function fit\_and\_score\_boosted\_trees.**\

*#-------- test\_implementation*

*# A function that tests your fit\_and\_score\_boosted\_trees function using three test sets.*

*# Input:*

*# fit\_and\_score\_boosted\_trees (your implementation of the boosting function)*

*# Returns:*

*# None*

​

**def** test\_implementation(fit\_and\_score\_boosted\_trees):

*# Iterate over test cases*

**for** i **in** range(1,4):

*# Load train & test data*

data\_train = np.loadtxt('testcases/challenge\_testcase\_' **+** str(i) **+** '\_train.txt', delimiter=',')

data\_test = np.loadtxt('testcases/challenge\_testcase\_' **+** str(i) **+** '\_test.txt', delimiter=',')

​

*# Split label and instances*

y\_train = data\_train[:,**-**1]

x\_train = data\_train[:,0:**-**1]

​

y\_test = data\_test[:,**-**1]

x\_test = data\_test[:,0:**-**1]

​

*# Run boosting function*

**print** 'Test case', i, ':', fit\_and\_score\_boosted\_trees(x\_train, y\_train, x\_test, y\_test, 10, 2)

# HW8

## Problem 1: Image Processing Revisited

In this problem we revisit applications of classification, with the purpose of comparing the performance of support vector classifiers with other classifiers we have learned. We'll begin with the aeriel vegetation detection problem from Homework #7.

The data is contained in dataset\_1.txt and dataset\_2.txt (you are encouraged to use the datasets from Homework #7 as well). The first two columns of the data contains the latitude and longitudes of randomly sampled locations in the satellite image, and the last column contains a label indicating whether the location contains vegetation (1 denotes the presence of vegetation and 0 denotes otherwise). The task is to, again, identify the vegetation regions in the image.

* Compare the result of using support vector classifiers to perform classification against results obtained from other models you have learned. Which model is more appropriate for the general task of vegetation detection in aerial images (do not restrict yourself to which model performs better on just these two datasets)? Which model is more appropriate for other types of image processing (hand-writting digit classification for example) Your comparison should be both **qualitative** and quantitative.

**Hint:** For your analysis, it's vital to consider the differences between the ways in which each of these models perform classification. These differences can be gauged by looking at the differences between the decision boundaries drawn by the models and **how** these boundaries are determined.

* Are there any obvious draw backs to support vector classifiers as we have presented them to you? What might be some intuitive ways to address these draw backs?

Again, we provide you with a function plot\_decision\_boundary to visualize the decision boundary of a classifier.

*#-------- plot\_decision\_boundary*

*# A function that visualizes the data and the decision boundaries*

*# Input:*

*# x (predictors)*

*# y (labels)*

*# model (classifier)*

*# poly\_flag (fits quadratic model if true, otherwise linear)*

*# title (title for plot)*

*# ax (a set of axes to plot on)*

*# Returns:*

*# ax (axes with data and decision boundaries)*

​

**def** plot\_decision\_boundary(x, y, model, title, ax, bounds=(0, 1), poly\_flag=False):

*# Plot data*

ax.scatter(x[y **==** 1, 0], x[y **==** 1, 1], c='green')

ax.scatter(x[y **==** 0, 0], x[y **==** 0, 1], c='white')

*# Create mesh*

interval = np.arange(bounds[0], bounds[1], 0.01)

n = np.size(interval)

x1, x2 = np.meshgrid(interval, interval)

x1 = x1.reshape(**-**1, 1)

x2 = x2.reshape(**-**1, 1)

xx = np.concatenate((x1, x2), axis=1)

​

*# Predict on mesh points*

**if**(poly\_flag):

quad\_features = preprocessing.PolynomialFeatures(degree=2)

xx = quad\_features.fit\_transform(xx)

yy = model.predict(xx)

yy = yy.reshape((n, n))

​

*# Plot decision surface*

x1 = x1.reshape(n, n)

x2 = x2.reshape(n, n)

ax.contourf(x1, x2, yy, alpha=0.1, cmap='Greens')

*# Label axes, set title*

ax.set\_title(title)

ax.set\_xlabel('Latitude')

ax.set\_ylabel('Longitude')

**return** ax

## Problem 2 (Optional): Classification Competition

This problem will involve an class-wide model building competition, where you will compete with each other in building a prediction model for cancer diagnosis. The results will be displayed live on a public leaderboard. The competition begins on Nov 9th, 3:00pm, and end on Nov 16th, 11:59pm.

Please access the following link for all relevant details (data set, submission instructions, evaluation metric, leader board, etc.):<https://inclass.kaggle.com/c/harvard-data-science-course-competition>

**Reward:** The top 20% of students on the leaderboard will each receive one bonus point to apply to an homework score of their choice (meaning you can turn a homework score of 2 to a 3, 3 to a 4, 5 to a 6 etc).

## Challenge Problem: Meta Learning

In the problem, you are provided with 10 different previously trained prediction models for a spam classification task. The task is to investigate how can one combine these models into a single meta classification model (without retraining the individual models) that performs better than each of the individual ones?

The data for this problem is provided in the files dataset\_5\_train.txt and dataset\_5\_test.txt. Each row of these files is an email described by 57 attributes, and the last column is 1 if the email is spam, and 0 otherwise.

The prediction models are provided in the file models.npy and can be loaded into an array by executing:

models = np.load('models.npy')

As before, you can make predictions using the ith using:

model[i].predict(x\_test)

and score the model using:

model[i].score(x\_test, y\_test)

The baseline for this task is a simple combination strategy that takes a majority vote from the individual prediction models.

**Any reasonable model that performs better than the baseline model on the test set will receive full credit.**

There are many intuitive ways to combine these 10 models into one; a more sophisticated approach is called "mixture of experts". In this problem, we are not requiring you to implement any particular approach.