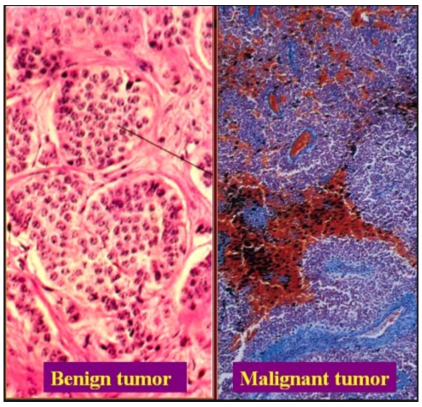
Breast Cancer Prediction Using Least Squares

10 points



For this problem, you will develop models using the least squares method to predict whether a tumor is malignant **M** (cancerous / deadly) or benign **B** (non-cancerous / safe). Models similar to these could help doctors determine if a person is at risk for having cancer and consequently detect and treat the cancer earlier.

A tumor is a mass of abnormal tissue. Malignant and benign tumors have different cell growth characteristics (See the image to the right for an example). Some of the important tumor properties include the *radius* and the *texture* among others. X-ray imaging and biopsies (*examining a small sample of the tumor under a microscope*) can be used to determine these characteristics.

You will be given a large data set containing hundreds of patients along with properties of their tumors. You will solve least squares problems with this data. You will then use the generated models to predict whether patients in another set have malignant **M** or benign **B** tumors.

We will be using the Python Data Analysis Library (Pandas (http://pandas.pydata.org/)) for importing the data and producing visualizations.

What Information Do I Have? (click to view)

You have the following data to work with:

- There are 2 data files named *breast-cancer-train.dat* and *breast-cancer-validate.dat* residing on the class server. You will open these data files using Pandas (http://pandas.pydata.org/). If you are working locally, you can access these files here (/course/cs450-s19/file-version/dd9a1d98076c43c7f314f5733421d849d1acad38/data/breast-cancer-train.dat) and here (/course/cs450-s19/file-version/dd9a1d98076c43c7f314f5733421d849d1acad38/data/breast-cancer-validate.dat). (See *What Do I Need To Do?*)
- You are given a list called `labels` that contains names (type: string). This list should be used when loading the data files with Pandas. The names will then become your column headers in the Panda data set.
- You are given a second list called `subset_labels` that contains the names of 4 columns you will use when creating a quadratic least squares representation. (See *Least Squares Theory* and *What Do I Need To Do?*)
- Lastly, you are given a handle to a function that will create a bar graph. This function is called `bar_graph`. (See *What Do I Need To Do?*)

Here is a quick example of a Pandas `DataFrame` class object called `your_data`. This will be the type of object returned from Pandas when opening your *.dat* files. These object are similar to a numpy 2D-array with a feature that you can access entries using the column and indices associated with each (e.g. `your_data['Malignant/Benign'][0]`). Or you can access by using the method `your_data.values[][]`. A snapshot of the data set is shown below. You can see all the columns and indices by running the commands `your_data.columns` and `your_data.index`.

You will notice that the first column is called *patient ID*. For each patient there is an entry in the 'Malignant/Benign' column indicating whether their tumor was malignant or benign. The remaining 30 columns give characteristics of the tumor.

	patient ID	**Malignant/Benign**	**radius (mean)**	**radius (stderr)**	**radius (worst)**	**texture (mean)**	**texture (stderr)**	**te (w**
0	842302	M	17.990	10.38	122.80	1001.0	0.11840	0.2
1	842517	M	20.570	17.77	132.90	1326.0	0.08474	0.0
2	84300903	M	19.690	21.25	130.00	1203.0	0.10960	0.
3	84348301	M	11.420	20.38	77.58	386.1	0.14250	0.2
4	84358402	M	20.290	14.34	135.10	1297.0	0.10030	0.
5	843786	M	12.450	15.70	82.57	477.1	0.12780	0.
6	844359	M	18.250	19.98	119.60	1040.0	0.09463	0.
7	84458202	M	13.710	20.83	90.20	577.9	0.11890	0.
8	844981	M	13.000	21.82	87.50	519.8	0.12730	0.
9	84501001	M	12.460	24.04	83.97	475.9	0.11860	0.2

Least Squares Theory (click to view)

Some essential theory for this problem is given below.

Consider an example set of data D^{7x2}, which has 7 rows and 2 columns and can be used to predict a result (in our case this will be using the tumor characteristics of each patient to predict if the patient has a benign or malignant tumor). D looks like,

$$D = \begin{bmatrix} d_{0,0} & d_{0,1} \\ d_{1,0} & d_{1,1} \\ \vdots & \vdots \\ d_{6,0} & d_{6,1} \end{bmatrix}$$

If you have a set of results $\boldsymbol{b}^{7x1}$ and would like to know how the data D can be used to best approximate the solution $\boldsymbol{b}$. The least squares formulation serves this purpose by solving $A\boldsymbol{w} = \boldsymbol{b}$, where A and $\boldsymbol{b}$ are known and A is some representation of the data D.

A **linear representation** of D would be,

$$A = \begin{bmatrix} d_{0,0} & d_{0,1} \\ d_{1,0} & d_{1,1} \\ \vdots & \vdots \\ d_{6,0} & d_{6,1} \end{bmatrix}$$

A **quadratic representation** of D would be,

$$\tilde{A} = \begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,0}^2 & d_{0,1}^2 & d_{0,0} \times d_{0,1} \\ d_{1,0} & d_{1,1} & d_{1,0}^2 & d_{1,1}^2 & d_{1,0} \times d_{1,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{6,0} & d_{6,1} & d_{6,0}^2 & d_{6,1}^2 & d_{6,0} \times d_{6,1} \end{bmatrix}$$

Once you have construct A or $\tilde{A}$, Singular Value Decomposition can be used to solve the system for $\boldsymbol{w}$.

Then, given a new set of data $E^{n,2}$ and the $\boldsymbol{w}$ just solved for, a predictive result can be solved for by putting E in the same form as either A or $\tilde{A}$ and performing a matrix-vector multiplication.

Lastly, the example given here had 2 columns, but least squares applies for problems with an arbitrary number of rows and columns, although one needs more rows than columns. For the problem given in this homework, you will be asked to create a quadratic representation with 4 columns of data. The matrix $\tilde{A}$ will then have the following form:

$$\tilde{A} = [\boldsymbol{\alpha}_0,\ \boldsymbol{\alpha}_1,\ \boldsymbol{\alpha}_2,\ \boldsymbol{\alpha}_3,\ \boldsymbol{\alpha}_0^2,\ \boldsymbol{\alpha}_1^2,\ \boldsymbol{\alpha}_2^2,\ \boldsymbol{\alpha}_3^2,\ \boldsymbol{\alpha}_0\boldsymbol{\alpha}_1,\ \boldsymbol{\alpha}_0\boldsymbol{\alpha}_2,\ \boldsymbol{\alpha}_0\boldsymbol{\alpha}_3,\ \boldsymbol{\alpha}_1\boldsymbol{\alpha}_2,\ \boldsymbol{\alpha}_1\boldsymbol{\alpha}_3,\ \boldsymbol{\alpha}_2\boldsymbol{\alpha}_3]$$

Where each $\boldsymbol{\alpha}_i$ represents a column.

What Do I Need To Do? (click to view)

You will solve two least squares problems, one using the **linear representation** and another with a **quadratic representation.** See *Least Squares Theory* for clarifications of what a linear or quadratic representation is.

1. Use Pandas to open the data sets *breast-cancer-train.dat* and *breast-cancer-validate.dat*:

```
import pandas as pd

your_data = pd.io.parsers.read_csv("data_to_read.dat", header=None, names=la
bels)
```

2. Take a look at some the data using plot commands from Pandas.
   - Generate a histogram of one of the data columns using `your_data["an interesting column"].hist()`.
   - Generate a plot, `your_data["an interesting column"].plot()`.
   - For both the histogram and plot, include titles, x-labels, and y-labels.
3. Construct linear and quadratic least squares representations of the data in *breast-cancer-train.dat* and *breast-cancer-validate.dat*. (i.e. 4 matrices). For the quadratic representation, you will only use a subset of the data available from *breast-cancer-train.dat* and *breast-cancer-validate.dat*. You are

given a list called `subset_labels` that contains the names of 4 columns that you should use for this purpose.

    - Call the linear least squares representation for *breast-cancer-train.dat* `A_linear`, this matrix will be graded for partial credit.
    - Call the quadratic least squares representation for *breast-cancer-train.dat* `A_quad`, this matrix will be graded for partial credit.

4. Construct a right-hand side vector $\boldsymbol{b}$ for both data sets. To create $\boldsymbol{b}$, make a numpy 1D-array the same size as the *Malignant/Benign* column of your data set and set each entry to 1 if the patient's tumor was malignant, otherwise set it to -1.
    - Call the right-hand side vector for *breast-cancer-train.dat* `b`, this will be graded for partial credit.
5. Solve for the linear and quadratic representation weights using the SVD of your A matrices built from the *breast-cancer-train.dat* data set. You should use your $\boldsymbol{b}$ built from the *breast-cancer-train.dat* data set as well.
    - Call the weights for the linear representation `weights_linear`. This will be graded for partial credit.
    - Call the weights for the quadratic representation `weights_quad`. This will be graded for partial credit.

    Note: You should not use `scipy.linalg.lstsq` or any similar function to solve the least squares problem. *Note:* Pass `full_matrices=False` to `numpy.linalg.svd` to obtain the reduced SVD. Otherwise computing the factorization will take longer than necessary.

6. See how well your weights predict whether a tumor is malignant or benign by performing matrix-vector multiplication of the linear and quadratic representations of the *breast-cancer-validate.dat* data set with `weights_linear` and `weights_quad`. This will produce a prediction vector $\boldsymbol{p}$. For each entry i of $\boldsymbol{p}$, if $p[i] > 0$, we will assume the ith person (probably) has a malignant tumor. If $p[i] \leq 0$ then the ith person (probably) has a benign tumor.
    - Compare your $\boldsymbol{p}$ for the linear and quadratic representation to the actual $\boldsymbol{b}$ (Use the $p[i] > 0, p[i] \leq 0$ rule given above).
    - Store the number of false-positives and false-negatives for both representations. A false-positive is when your model indicated that a tumor was malignant, when it was not. A false-negative is when your model fails to correctly identify a tumor as malignant.
    - Call the `bar_graph(fp_linear, fn_linear, fp_quad, fn_quad)` function and give it the number of false-positives *fp* and false-negatives *fn* for each representation.

All done? You've now gained some hands-on experience with an important workhorse of data science!

INPUT:

- `labels`: A list of strings which label the features in the data frame. You should include all these in your linear least-squares model.
- `subset_labels`: A list of strings indicating the names of the features which to include in the quadratic model.
- `bar_graph(fp_linear, fn_linear, fp_quad, fn_quad)`: A function to plot a bar graph of error statistics.

OUTPUT:

- b : The right-hand side to your least-squares problem
- A_linear : The matrix for your linear least-squares model
- weights_linear : The solution weights vector for your linear model
- A_quad : The matrix for your quadratic least-squares model
- weights_quad : The solution weights vector for your quadratic model

Starter code (click to view)

Answer*

```
#  ----------------------------------------------------------
#
#  THIS CODE IS INCOMPLETE! BUT MAY HELP WHEN GETTING STARTED
#
#  ----------------------------------------------------------
import numpy as np
import numpy.linalg as la
import pandas as pd
import matplotlib.pyplot as plt

#  Read in the data files
train = pd.io.parsers.read_csv("breast-cancer-train.dat", header=None, na
validat = pd.io.parsers.read_csv("breast-cancer-validate.dat", header=Non

#  Produce a Pandas histogram and plot (fill in appropriately)
plt.figure(0)
plt.title("Histogram of Cancer Cell's Radius vs Patients number")
train[labels[4]].hist()
plt.xlabel("Index of patients")
plt.ylabel(labels[4])
plt.figure(1)
train[labels[4]].plot()
```

Press F9 to toggle full-screen mode. Set editor mode in user profile (/profile/).

Overall grade

The overall grade is 100%.

Autograder feedback

The autograder assigned 7/7 points.

Your answer is correct.

Here is some feedback on your code:

- 'b' looks good
- 'A_linear' looks good
- 'A_quad' looks good
- 'weights_linear' looks good
- 'weights_quad' looks good
-
- Execution time: 2.1 s -- Time limit: 20.0 s

Your code ran on relate-01.cs.illinois.edu.

Human feedback

The human grader assigned 3/3 points.

Your code produced the following plots:

Figure0

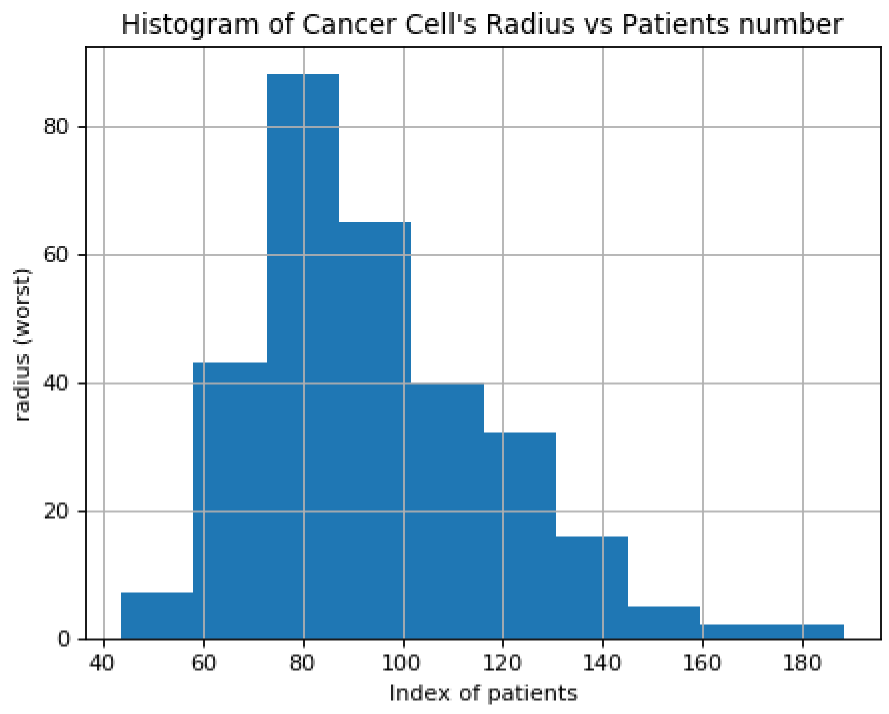


Figure1

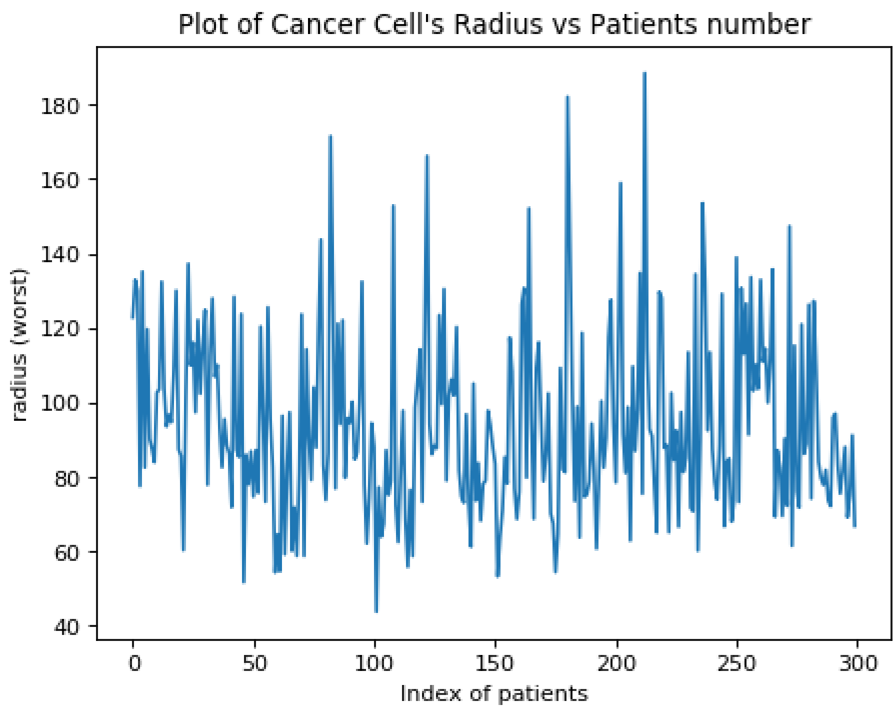
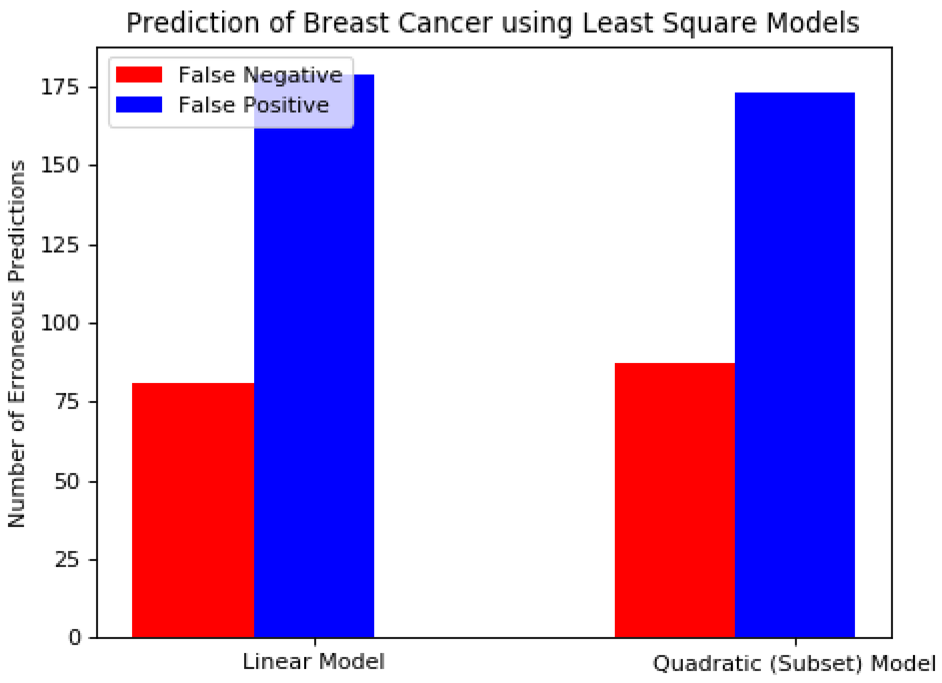


Figure2



The following code is a valid answer:

```
#  Correct Code
import numpy as np
import numpy.linalg as la
import pandas as pd
import matplotlib.pyplot as plt

#  Load the data
tumor_data = pd.io.parsers.read_csv("breast-cancer-train.dat", header=None, names=l
validate_data = pd.io.parsers.read_csv("breast-cancer-validate.dat", header=None, n

#  Construct some an interesting histogram and plot
plt.figure(0)
tumor_data["radius (mean)"].hist()
plt.title("This is the radius (mean) histogram")
plt.xlabel("tumor - radius (mean)")
plt.ylabel("number of tumors for each bin")
plt.figure(1)
tumor_data["radius (mean)"].plot()
plt.title("This is the radius (mean) plot")
plt.xlabel("patient number")
plt.ylabel("tumor - radius (mean)")

#  A Function for Constructing the Matrix Models
def construct_A_quadratic(A_linear, subset_labels, tumor_data):
    sublength = len(subset_labels)
    num_cross_terms = int(sublength*(sublength - 1)/2)
    num_cols = 2 * sublength + num_cross_terms
    A = np.zeros((A_linear.shape[0], num_cols))
    #  The Linear and Pure Quadratic Terms
    count = 0
    for i, col in enumerate(A_linear.T):
        if tumor_data.columns[2 + i] in subset_labels:
            A[:, count] = col
            A[:, count + 4] = col*col
            count += 1
    #  Fill-in the Cross-Terms
    #+ (i.e. x1*x2, x1*x3, x1*x4, x2*x3, ... )
    alpha = 0
    col = 2 * sublength
    while col < A.shape[1]:
        for idx in range(alpha + 1, sublength):
            A[:, col] = A[:, alpha]*A[:, idx]
            col += 1
        alpha += 1
    #  Return The Matrix
    return(A)

#  Construct the RHS
b = tumor_data["Malignant/Benign"].values
b = (b == "M").astype(np.float64)*2 - 1
v = validate_data["Malignant/Benign"].values
v = (v == "M").astype(np.float64)*2 - 1


#  Construct the Linear Models
#+ A: is the test data
#+ B: is the validation data
```

```
A_linear = np.float64(tumor_data.values[:, 2:])
B_linear = validate_data.values[:, 2:]
#  Construct the Quadratic Models
#+ A: is the test data
#+ B: is the validation data
A_quad = construct_A_quadratic(A_linear, subset_labels, tumor_data)
B_quad = construct_A_quadratic(B_linear, subset_labels, validate_data)


def solve_lstsq(A, b):
    U, sigma, VT = la.svd(A, full_matrices=False)
    return VT.T @ ((U.T @ b)/sigma)

#  Solve the Linear Model
weights_linear = solve_lstsq(A_linear, b)

#  Solve the Quadratic Model
weights_quad = solve_lstsq(A_quad, b)

#  How Well Does the Linear Model Do?
p_linear = B_linear.dot(weights_linear)
p_linear[p_linear >  0] =  1
p_linear[p_linear <= 0] = -1
fp_linear = len(np.where(p_linear > v)[0])
fn_linear = len(np.where(p_linear < v)[0])

#  How Well Does the Quadratic Model Do?
p_quad = B_quad.dot(weights_quad)
p_quad[p_quad >  0] =  1
p_quad[p_quad <= 0] = -1
fp_quad = len(np.where(p_quad > v)[0])
fn_quad = len(np.where(p_quad < v)[0])

#  Generate a Bar Graph Plot of the False Pos/Neg Data
bar_graph(fp_linear, fn_linear, fp_quad, fn_quad)
```