# SYSC 4906 Introduction to Machine Learning Fall 2020

Assignment 1

**Submission instructions: Please prepare a single Jupyter Notebook with all of your answers. Some answers will only require text while others require text+code+results. Please use the template solution available from the course GitHub repo under "Assignments/Assignment1".**

Q1) Calculate the gradient of the $f(x, y, z) = x^3z - 2xy^2 + 5z$ at (-3, -2, 1). What does this vector represent?

Q2) You are working at a COVID-19 testing center as a data scientist and are asked to determine how severe the symptoms of patients who come to get tested are. For a day, you record each person's symptom severity using a subjective rating on a 1-5 scale for. These are the results from that day:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 4 | 3 | 4 | 3 | 1 | 3 | 2 | 3 |

1. What is the <u>expected value</u> from this sample? Using an <u>unbiased estimator</u>, what is the sample <u>variance</u> and <u>standard deviation</u>?

After a long time of collecting this data, your statistician friend has determined the probability mass function, Pr(x), where x is severity rating. They also found the probability that a patient tested positive for COVID, given their symptom severity, Pr(+|x). They provide the table below, but you spilled coffee on Pr(2) and can't make out the value. You need to present these results to your boss!

| **x** | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **Pr(x)** | 0.20 | ?? | 0.30 | 0.30 | 0.05 |
| **Pr(+\|x)** | 0.10 | 0.15 | 0.20 | 0.40 | 0.70 |

2. What is Pr(2)?
3. Find the expected value and the variance for Pr(x).
4. Find the probability that a patient has a symptom severity of x=5, given that they tested **negative** for COVID-19. That is, find Pr(5|-). *Hint: Pr(+) can be found by summing over Pr(+|x)Pr(x), for all x. Pr(-|x) can be derived from Pr(+|x).*

Q3) Create a python notebook which loads the Kaggle Heart attack possibility dataset (https://www.kaggle.com/nareshbhat/health-care-data-set-on-heart-attack-possibility). This dataset has 13 features each and 2 classes of heart attack possibility: target: 0= less chance of heart attack; 1= more chance of heart attack. *Hint: look at the notebooks from Tutorials 2 & 3 for example code for achieving the steps below.*

a) Split the data, using 75% for training and 25% for test. Make sure you use stratified sampling.
b) Train and test a logistic regression classifier. How accurate is your classifier?
c) Repeat part b), but using only the age and resting blood pressure features from the dataset. Was the classifier accuracy impacted?
d) Using the 2-feature classifier from part c), create two subplots using the age and resting blood pressure features from the data set. *See Tutorial 3 Part 2 for similar plots.*
 i) On the first, plot the decision boundary and the training data. Use green for less chance (target==0) and blue for more chance (target==1).
 ii) On the second, plot the decision boundary and the test data. Use the same colours (blue/green), but highlight all misclassified test points (from either class) in red.

Q4) Linear regression. Download the file "Assig1Q3.csv" from the course GitHub repo under "Assignments/Assignment1". The first column represents the X values, while the second column represents the Y values.

1. Plot the data
2. We are going to use linear regression to fit a linear and a quadratic model to these data. **Without using sklearn.linear_model** (or other linear regression libraries), write your own python code to implement the least squares solution for linear regression. That is:
$$\beta = (X^TX)^{-1}X^Ty$$
3. Assuming the model $y = mx + b$, use your code to best-fit the parameters m and b to the data. Report your optimal parameter values.
*Hints:*
    a. *recall that you must create the 'augmented' feature vector X from the given x data (add a column of 1's).*
    b. *look at numpy.T(), numpy.matmul(), numpy.dot(), and numpy.linalg.inv()*
4. Plot your line of best fit on top of the data
5. Calculate the sum of square residuals, or mean squared error, as in:
$$MSE(\beta) = \sum_{i=1}^{N}(y - X\beta)^2$$
6. Assuming the model $y = ax^2 + bx + c$, repeat steps 2-4 using this new model (i.e. estimate the optimal values for a,b,c; report those estimates; plot the line of best fit; report the MSE).
7. Briefly discuss which model would you prefer for these data?
8. Why is best-fitting the second (quadratic) model still considered linear regression?

Q5) Create a Jupyter Notebook based on `Tutorial-3_ComparingMultipleClassifiers.ipynb` to use `make_classification` to create a linearly separable dataset, with 2 classes, 2 informative features, 1500 samples per class, using a class_sep=1.7, and a random_state of 5. Generate some random noise of the same shape as your feature data, drawn from a **standard normal distribution** (see `numpy.random`) and a random_state of 5.

1. Create four datasets: 1) no noise, 2) data + 0.5 * noise, 3) data + 1.0 * noise, and 4) data + 2.0 * noise. For all four datasets, plot the data, labelling each (sub)plot by the degree of noise added (i.e. 0, 0.5, 1.0, and 2.0)
2. For each dataset, create training and test data using a 70/30 train/test split
3. For each dataset, train and test an SVM classifier with a polynomial kernel with degree=2, and C=1.0. Report the test score for each. How does prediction accuracy change with noise level?
4. For a noise level of 0.5, train and test SVM classifiers using the following values for C: {0.001, 0.01, 0.1, 1, 10, 100}. Report the test accuracy for each. How does performance vary with C? Briefly describe what the `C` controls for sklearn.svc. *Hint: look at the documentation for sklearn.svc rather than the class notes here...*