1. **A discussion of the dataset**
   1. Where did it come from? Who published it?
      1. I acquired this dataset from a company called Optimus that does this sort of analysis for professional politicians. This set of data itself is originally from the L2 voterfile combined with some commercial data. Because of all the in-house processing at this point Optimus would be the source of the data.
   2. Who cares about this data?
      1. Anyone who is interested or invested in politics should be interested in this dataset and those similar to it. Voter turnout prediction software and analysis is a major industry in the political realm. Any professional political campaign will pay for this kind of analysis. Many companies perform these services through polling the current population, but those outcomes can be bias and inaccurate depending on the sample used, the questions asked, or even the company performing the analysis. Using machine learning algorithms on past voter turnout data is an objective way to approach this complex problem, that has the potential to get better results as it’s training data increases year after year.
2. **A discussion of the problem to be solved**
   1. Is this a classification problem? A regression problem?
      1. The question I’m asking is whether a voter will vote in a coming election or not. This problem is ultimately a binary classification problem that sorts potential voters into a prediction whether they will vote or not. However, I decided to solve this problem using Gaussian Process Regression for a number of reasons. While it’s nice to have a specific binary value in the end for voting or not voting it’s much more interesting to have a probability for how likely the voter is going to vote. One reason for this regressive approach is that with a probability in the future I could split these results into very unlikely, unlikely, likely, and very likely voters depending on their probabilities. Having a probability makes it easy to split the potential voters into further categories. Upon researching similar approaches to this problem this approach is called a supervised probabilistic classification algorithm.
   2. Is it supervised? Unsupervised?
      1. This problem is a supervised machine learning algorithm because the training data includes labels that shares each voter’s voting history. The algorithm uses these labels to make intelligent predictions for voters with similar backgrounds.
   3. What sort of background knowledge do you have that you could bring to bear on this problem?
      1. In this problem, it’s useful to have as much information about voting patterns historically as you can. I know, for instance that young voters that are younger than 22 weren’t able to vote in past general elections because they were not 18. I don’t really know how to solve this problem with the current data set because it seems a lot like a cold start problem to me. With additional data and testing I would be able to create a predictor for new and young voters based on their background information.
   4. What other approaches have been tried? How did they fare?
      1. Another method that has been used to predict voter turnout is decision trees or random forests method. In this approach one would use the training data to create a decision tree based on attributes from a random subset of the data. The goal is to create a tree that matches the sample so that one could easily traverse the tree and reach a yes or no leaf node and be able to predict the turnout for one particular voter. One obvious challenge with this approach is that it’s easy to overfit the data, capturing in too much noise and not getting a very good decision tree. One way to combat this overfitting is to make a “forest” of decision trees and randomly select one tree to use for each sample. This randomness helps to prevent overfitting because each individual tree is a different representation of the data. I don’t know how random forests fare in terms of their RMSE but I can imagine that there is quite a bit of variability with the outcome depending on how effective each individual decision tree can get a random voter to the correct outcome.
3. **A discussion of your exploration of the dataset**.
   1. Before you start coding, you should look at the data. What does it include?
      1. The data includes all of the following labels.
      2. It also has a section that shows the voter history from 2000 clear until 2014.



* 1. What patterns do you see?
     1. One of the most interesting pattern that I searched was the turnout rates of this same sample for each general election. I did see that every other election had a higher turnout than the election previous. This is pretty typical because there is a higher turnout for presidential elections than there is for a senator or house seat election. However, the surprising thing about this particular dataset was that the typical pattern for turnout was inverted with the presidential elections having a lower turnout than the primaries. I haven’t really figured out why this patter was the way it was. I would have assumed it to be opposite.
     2. I also saw that a voter was more likely to show up if he was quite a bit older with the mean around age 60.
     3. Another pattern that I thought was pretty interesting and that I was happy to see turn up in the predictions was that when considering ethnicity most ethnicities were fairly well balanced in this dataset leaning slightly towards Republican. This pattern was true among all ethnicities except Hispanics and African-Americans who leaned very heavily Democrat.
  2. Any visualizations about the data you deem relevant



1. **A clear, technical description of your approach.** This section should include:
   1. Background on the approach
      1. Gaussian Process Regression also known as kriging is used to infer continuous values from some given training data. Typically, the training data is put into a covariance matrix where a k function measures similarity between any two data points in the matrix. Because of the n^2 space complexity associated with this matrix it’s difficult to create for large data sets. One solution for large scale data is to select or create landmark entries that are representative of a specific cluster of data. This would allow the covariance matrix to be much smaller while still striving to get a representative sample of entries from the data. Given the covariance matrix and a set of test data points estimation of each point is a simple matter of performing a few linear algebra operations of the matrix, the training values, and two matrices that compare the test data to each landmark column of the covariance matrix. Gaussian Process Regression has many applications in statistics and machine learning. Although this algorithm result in continuous values, those values could be probabilities which can then be used to categorize your predictions. This application is called Probabilistic Classification and is the approach we’re using for this project.
   2. Description of the model you use
      1. This version of Gaussian Process Regression doesn’t use a specific model. The most modeling that takes place is when I split all the data points into different clusters. In order to cluster the data I used the sklearn.cluster library, and had to convert each of the data entries that I wanted to cluster into numerical data. In order to do that I created a map from the number to the term that each number represented. I converted partisanship, ethnicity, marital status, and education all into a numerical representation of those values. Ultimately it didn’t matter what values I used as long as it was a consistent mapping for the clustering algorithm.
   3. Description of the inference / training algorithm you use
      1. As mentioned above I used a Probabilistic Classifier made of a large-scale Gaussian Process Regression algorithm. In order to train my algorithm, I first had to format my data. The first thing I did was remove any unused data from my set. Ideally, I would use every entry but there were a number of columns that I either didn’t think would be relevant or didn’t want to focus on for this algorithm, some of these categories include is there an individual interested in music in the household, or is there an individual interested in NASCAR.
      2. Once I had the basic columns that I wanted to use I needed to convert each of those values into a numerical data point so that the sklean clustering algorithm could use a comparator to cluster the data. As mentioned above I used a couple of maps and list comprehension to accomplish these conversions. I also created a new column called that calculated the probability that one particular voter showed up to the past general elections.
      3. After cleaning the data, it was a simple matter of running it through the clustering algorithm. Once I had my clusters I created a representative landmark by taking one random entry in the cluster and setting its general election probability to the mean of it’s corresponding cluster.
      4. With the landmarks representing the data I was able to create the covariance matrix by simply running them through the k function that would compare two entries. One of the major improvements that I made over my lab in class was that I figured out how to parallelize the matrix creation using numpy meshgrids. It was one of the things that took the longest, but it was definitely worth the effort because I was able to run my entire algorithm in about 30 minutes, compared to several hours it would have taken before with the same number of landmarks.
   4. Description of how you partitioned your data into a test/training split
      1. Originally, I used all of my data as training data because I was trying to predict future turn out rather than improve upon the data that I already had. However, after the last lab I realized the huge benefit that creating test data for the RMSE formula. Once I was able to create that formula I was better able to tell which formula changes increased and decreased my accuracy. In order to make that split I decide to ask Optimus for the actual user data, I didn’t know if they would send it to me but I figured it would be worth a shot. If they didn’t send me the data I would have moved my prediction engine back to the previous election in order to simulate not knowing the future, while having the actual turnout numbers. Luckly…………………………………………………………………………………………………
2. **An analysis of how your approach worked on the dataset**
   1. What was your final RMSE on your private test/training split?
   2. Did you overfit? How do you know?
      1. Overfitting the data is always a concern when using Gaussian Process Regression. Because I’m using all of my data to both train and validate that could be a real issue. However, I think that I’m okay because of how I create my covariance matrix. My covariance matrix isn’t a basic random subset of the data, rather it tries to be a representative subset by using clusters. I don’t know how the clustering algorithm works but I do know that it does create different clusters each time. So as long as I’m creating these clusters each time while I tune my algorithm it’s not very likely that I will be overfitting the data.
      2. I think that there might be a greater risk in overfitting if the covariance matrix were a full comparison for all of the data points. That being said one solution to avoid overfitting in the future would be cross validation. Instead of using all of my data as training I could split it into training and testing data. Then I could use the testing data as a measure of how well the algorithm is doing with no way to actually overfit the data because it’s completely separate data.
   3. Was your first algorithm the one you ultimately used for your submission? Why did you (or didn't you) iterate your design?
      1. No, once I was able to calculate the RMSE, I iterated on this algorithm to tune the k matrix. In order to avoid overfitting I kept creating the landmarks each time, but that also made it very difficult to know if it was my changes to the k matrix that were making a difference or I just happened to have better landmarks on this iteration. The main progress I did make while iterating was that I was able to vectorize the matrix creation process. This led to drastic increases in performance.
   4. Did you solve (or make any progress on) the problem you set out to solve?
      1. Obviously, I have a solution to this problem, but it’s not the solution. I was able to make a lot of progress towards a better solution, but there is still a lot of tinkering and adjusting that would have to happen in order for this algorithm to produce consistently accurate results. As with many machine learning algorithms I think that with more training data, and more iterations of tweaking I would be able to get this algorithm to work pretty effectively.