RoboCoach: Technical Implementation and Ethical Considerations

Daniel Alpert, Sam Oh

December 14, 2017

We, Sam Oh and Daniel Alpert, affirm our awareness of the standards of the Harvard College Honor Code.

1 Abstract

The focus of this project is twofold: first, create an automated coaching recommender system that will recommend training for the next day given the previous day of training. This primarily technical problem consisted of two main parts: (1) the naive Bayes classifiers, and (2) the k-nearest neighbors algorithms. The first set of algorithms allowed us to numerically predict subjective running metrics from a set of written running logs, namely whether a day was a workout or not and also the difficulty of a day. These predictions were about 92.6% accurate for workouts and were about 85% accurate at predicting difficulty scores within 1 of the true score. These running metrics were then used to train the K-nearest neighbors algorithm that predicted the training for the next day based on the average of the k closest data points to the input. Through numerous trials, we converged on an optimal k value of about 25 which ended up with an R^2 value around 0.1 for the difficulty prediction, and an accuracy score of about 0.786 for the workout prediction. Finally, we considered two groups of ethical considerations related to the technical portion of our project: the ethical considerations of (1) our RoboCoach implementation and (2) a future, perfected iteration of RoboCoach. Through both the technical implementation of RoboCoach as well as an ethical analysis of the recommender system, we were able to successfully combine course topics to create and analyze an interesting problem.

2 Introduction

Both Sam and Daniel ran competitively for many years. With competitive running, it is crucial to have a coach that is able to assign mileage, namely how many miles to run on a given day along with the type of run to do. For example, a coach may assign 14 miles and a workout (meaning track repetitions, hill sprints, fartleks, etc.) or 8 miles and an easy run. Given the quantitative nature of the sport, we asked the following question: can we automate the coaching process by creating a recommender system that provides these details? This sort of system could potentially provide elite level coaching to people who may not have had access to such resources in the past. Moreover, this system could eventually learn more about the athlete and provide perfectly catered training based on sound evidence and data.

For this project, we created a coaching recommender system, RoboCoach, using two naive Bayes classifiers to classify the running logs of four past Division One Track & Field athletes and two k-nearest neighbors algorithms in order to recommend the amount to run as well as the type of run (workout or not workout) for the next day. The data we gathered had information about the distance run in a given day as well as a text field or log full of subjective information about how the athlete felt and also what type of run the athlete performed. Using the naive Bayes classifiers, we classified each day as a workout day or a non-workout day and also provided a difficulty score for each day from a scale of one to five, where a one represents a very easy day and a five represents a very difficult day. Once we classified every day of training in each of the logs, we used this information for any given day X as well as the following day $X + 1$ in order to train a k-NN classifier that allows us to prescribe training for day $X + 1$ given the training information done on day X. For an explicit representation of this data, see Figure 2: Training Data. Based on this data, the algorithm returns a recommendation of the distance to run the next day as well as the type of run (workout or not workout).

Overall, both the naive Bayes classifiers and the KNN algorithm worked accurately to produce a viable, and reasonable recommendation. The naive Bayes classifier accurately classified workouts about 93.6% of the time and accurately predicted difficulty scores within 1 of the true score about 85% of the time. Through numerous trials, we converged on an optimal k value of about 25 which ended up with an R^2 value around 0.1 for the difficulty prediction, and an accuracy score of about 0.786 for the workout prediction.

Given the ethical implications of creating such a system, we included a discussion on some of the ethical considerations that follow from the building of this system. In addition to the ethical implications of RoboCoach, the recommender system we created, we also examined the ethical implications of an advanced, optimal, future version of RoboCoach given that this advanced system engenders new ethical considerations that do not exist for the system we created.

3 Technical Problem Specification

Our goal for this project is to create an intelligent system that can take in the past day (or days) of training for a high-level runner and return a suggestion for the next day's training. The data that we will take in is how many miles were run the previous day as well as a description of how the training went (details of the run/workout, how the athlete felt, specifications of the number of repetitions done for workouts, miscellaneous thoughts). The description of how the training went will be in a text format such that the user can input this description in written prose. From this text description we want to elicit two different scores—one score that specifies how hard that day of training was (based on what the athlete wrote about the training) and a binary classification of whether or not that day was a workout (defined as doing repetitions, hills, cutdowns, or fartleks as opposed to just a normal run). These three covariates—distance run, difficulty, and whether the day was a workout—are what we believe are the most important factors in what goes into formulating and adjusting the next day of training for an athlete. We want to use these three variables from day X to make a training recommendation for day $X + 1$.

4 Approach

To classify both a difficulty score and whether the training was a workout or not, we trained two different supervised naive Bayes classifiers. The initial data in the training log, as shown in Figure 1, was messy — there was a lot of extra noise in the data set. First, we cleaned the data such that each log only had the date, the distance run, and notes for each day as seen in Figure 2. In the training process, the supervised naive Bayes classifiers looked at the frequency of each word in an input text field as well as the classification of that text (in this case, workout or not workout and difficulty score). From these inputs, the classifier generates and updates the likelihood (as a percentage) that a group of text should be classified into each specific class based on a certain collection of words. This is a supervised model (requires pre-labeled data), so we had to manually label some of our text data for both a workout classification and difficulty score before training this model. In the labeling phase, both Daniel and Sam calibrated the labeling in order to minimize labeling error prior to labeling the actual data set. We also split the data set into a training and testing data set, all of which was labeled, in order to measure the effectiveness of the naive Bayes classifier.

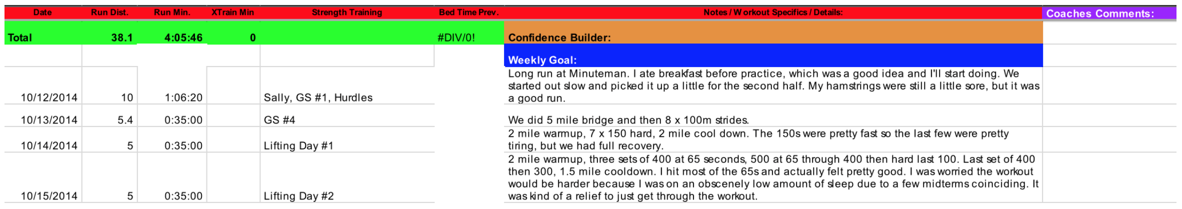
To suggest the next day's training, we will use a *k*NN classifier. This operates under the assumption that similar days of training yield similar training the next day. In other words, there exist patterns in the relationship between any two training days. A certain type of difficult day is usually followed by an easy day, whereas an easy day may typically be followed by a difficult day. With a *k*NN classifier, we can use our three variables to find the *k* nearest (most similar) other data points using a metric like Euclidean distance, take the average of the next day's training for those *k* closest points, and prescribe that average as the next day's training for our input point. This method is also supervised, but the next day's training is used as our "label" instead of manually labeling data points. In other words, the label for a given day's training is simply the training data from the following day, as seen in Figure 2.

5 Data

Runners on the Cross Country and Track & Field teams at Harvard are required to keep training logs containing extensive information about their day-to-day lives. These logs are shared with each athlete's coach so the coach can use the information to prescribe the best possible personalized training each day. These metrics include running specific ones—distance run, time run, workout specifics, cross training minutes, and strength/lifting training, but also more personal information—bed time the previous night, hours slept, feelings on the workout (mentally, physically, emotionally), and weekly goals.

A few days of an average training log look like this:

Figure 1: Training Log



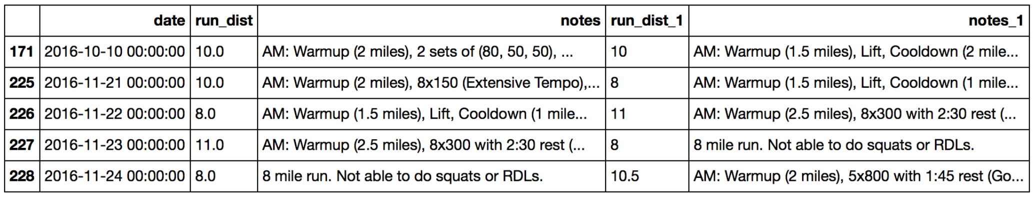
Date	Run Dist.	Run Min.	XTrain Min	Strength Training	Bed Time Prev.	Notes / Workout Specifics / Details:	Coaches Comments:
Total	38.1	4:05:46	0		#DIV/0!	Confidence Builder:	
						Weekly Goal:	
10/12/2014	10	1:06:20		Sally, GS #1, Hurdles		Long run at Minuteman. I ate breakfast before practice, which was a good idea and I'll start doing. We started out slow and picked it up a little for the second half. My hamstrings were still a little sore, but it was a good run.	
10/13/2014	5.4	0:35:00		GS #4		We did 5 mile bridge and then 8 x 100m strides.	
10/14/2014	5	0:35:00		Lifting Day #1		2 mile warmup, 7 x 150 hard, 2 mile cool down. The 150s were pretty fast so the last few were pretty tiring, but we had full recovery.	
10/15/2014	5	0:35:00		Lifting Day #2		2 mile warmup, three sets of 400 at 65 seconds, 500 at 65 through 400 then hard last 100. Last set of 400 then 300, 1.5 mile cooldown. I hit most of the 65s and actually felt pretty good. I was worried the workout would be harder because I was on an obscenely low amount of sleep due to a few midterms coinciding. It was kind of a relief to just get through the workout.	

The data used in this project is composed of our the training logs of the authors as well as those of two really close friends in this project (both of whom provided verbal and written consent). This gave us 11 years (33 track seasons and 11 summer trainings) of data between 4 different athletes (there are three separate track seasons as well as a single summer season in any given year). We should note that all of the data is from high-level male runners aged 18-22, so the training data in this set is highly homogeneous.

The data includes many variables, many of which only have marginal impacts on the next days training, so we cleaned the data to include only the two most important aspects—the running distance (weekly mileage is considered the most important single metric in distance running) and the notes because of the wide variety of information they carry.

Because the aim of the model is to take in the previous day's data and recommend training for the current day, the algorithm could only use the running logs that had both running distance and notes filled out for both that day (serving as the x variables in kNN) and the *next* day (the y variables for kNN). Any days that did not have both that day and the next filled out were removed from the data set. A snapshot of the resulting data set is shown below, where "run_dist" is the current day's distance run, "notes" is that day's notes/workout details, and the variables ending in "_1" are the same but for the next day:

Figure 2: Training Data



	date	run_dist	notes	run_dist_1	notes_1
171	2016-10-10 00:00:00	10.0	AM: Warmup (2 miles), 2 sets of (80, 50, 50), ...	10	AM: Warmup (1.5 miles), Lift, Cooldown (2 mile...
225	2016-11-21 00:00:00	10.0	AM: Warmup (2 miles), 8x150 (Extensive Tempo),...	8	AM: Warmup (1.5 miles), Lift, Cooldown (1 mile...
226	2016-11-22 00:00:00	8.0	AM: Warmup (1.5 miles), Lift, Cooldown (1 mile...	11	AM: Warmup (2.5 miles), 8x300 with 2:30 rest (...
227	2016-11-23 00:00:00	11.0	AM: Warmup (2.5 miles), 8x300 with 2:30 rest (...	8	8 mile run. Not able to do squats or RDLs.
228	2016-11-24 00:00:00	8.0	8 mile run. Not able to do squats or RDLs.	10.5	AM: Warmup (2 miles), 5x800 with 1:45 rest (Go...

The final dataset has 1972 observations, meaning there are 1972 days where running distance and notes were filled out both that day and the next day.

6 Naive Bayes

The first aspect of our project was translating our "notes" into both a difficulty and workout score. To do this we used the `naive_bayes` module in the `sklearn` package in Python, which has great functionality for a Multinomial naive Bayes model. As previously mentioned, in naive Bayes, all of the words in the dataset are turned into word vectors, where each column represents a word, and the corresponding values indicate how many times that word appeared in a specified chunk of text. The equation for naive Bayes is:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

where c is the class, x is the feature (word), $P(c|x)$ is the posterior probability, $P(x|c)$ is the likelihood, $P(c)$ is the class prior probability, and $P(x)$ is the predictor prior probability.

For example, let's say there are 10 groups of text, 5 of which are workouts and 5 of which are not. The word "intense" shows up in 6 of the groups of text, 4 times as a workout and 2 times as not. When the classifier sees a new group of text and sees the word "intense" we would want to know P(workout—intense), namely, the probability of a being a workout or not given the word "'intense". This is a conditional probability distribution based on the two classes of workout: workout and not workout. This could be solved by using the bayes rule equation shown above as follows:

$$P(workout|intense) = \frac{P(intense|workout)P(intense)}{P(workout)} = \frac{\frac{4}{5}\frac{6}{10}}{\frac{5}{10}} = 0.96$$

Similarly:

$$P(noworkout|intense) = \frac{P(intense|noworkout)P(intense)}{P(noworkout)} = \frac{\frac{2}{5}\frac{6}{10}}{\frac{5}{10}} = 0.48$$

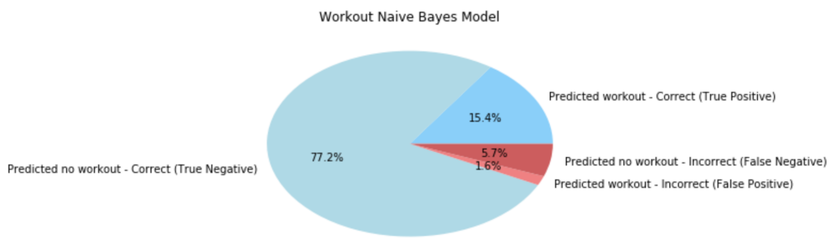
To find the probabilities of each, normalize both of these values so that all probabilities add to 1. After normalizing the values, the classifier believes that after seeing the word "intense," there is a $\frac{2}{3}$ probability the class is workout and a $\frac{1}{3}$ probability it is not a workout. This same process is repeated for all of the words in a group of text for each potential class, multiplied together and summed according to the class, and normalized across each potential class to find the likelihood that a group of text falls into a specific class.

As is demonstrated, all of the expressions on the right side of the equation can be found using the word vectors and labels from our data.

6.1 Workout

The naive Bayes workout model looks at the text and determines whether that given day was a workout or not. In order to allow the model to train correctly, we read the notes of 407 observations and manually labeled them as workout (1) or not (0). The naive Bayes model was trained on this data, achieving on average a 0.926 success rate in interpreting whether the text described a workout. This remarkably good result is likely due to specific words that are almost always used only when talking about one class. For example "workout," "reps," "repeats," and "##x##" (indicating some amount of reps done for some distance) are generally only used when describing workouts. Words like "miles," "mileage," and "bridge" (often describing a running route done in Cambridge) indicate a normal run, not a workout. Naive Bayes classifiers work best when this is the case—certain words are total tip-offs for which class the text belongs to.

Figure 3: Naive Bayes Model for Workouts



6.2 Difficulty

The same process was employed for classifying a "difficulty" score. With difficulty, the days are labeled according to how the runner felt, with 1 being "great" and 5 being "terrible." A day where a runner felt like the training was easy, encouraging, or "smooth" would be closer to a 1, while days where a runner was "falling apart," felt sick, or felt sore were closer to 5. Days that show no indication of feeling (i.e "Ran 6 miles") were given a score of 0. The variable name of "'difficulty" may be a misnomer because a day that was very physically difficult, but the athlete felt really good doing was given a lower score (i.e. less difficult) and a day that felt incredibly challenging but was in reality not too difficult (think an easy run but the runner was sore and sick) was given a high score (i.e. more difficult). As evidenced below, our manual classifications are an inexact science, but both labelers calibrated scores so that labeling of text was nearly the same between the two labelers. If not the same, at the very least, each label in the classifications are within 1 of its "true" difficulty.

Table 1: Example for Each Difficulty Score

Score	Text
1	AM: 4 mile run, PM: Warmup (3 miles), 2 sets of (120, 120, 120, 100, 80), Cooldown (1.5 miles). Speed development **felt pretty good**. All the reps **felt smooth**.
2	Warmup (2 miles), 4 sets of 2x200 (First set in 27, Second set in 26, Last two sets in 25) with 1:00 between reps and increasing rest between sets (3, 4, 5). It **went pretty well**. The second 200 on the last couple sets was **a little tough** but all the other **reps were easy**. **Felt smooth**.
3	10 mile run in 64 minutes. Legs were **pretty sore to start** but **got better** as the run went on.
4	AM: Warmup (1.5 miles), 6x3x75m hills with 2:00 between reps and 5:00 between sets, Cooldown (2 miles); PM: 3 mile shake out. Hill was a little steeper than Danehy but not as steep as Belmont. **Tough on the legs**.
5	AM: 3 miles; PM: Warmup (2.5 miles), 4x400, 300 off 5:00 (Goal: 56; Actual: 56, 55, 58, 64, 45), Cooldown (2.5 miles). Another **bad workout**. Legs **felt dead** before the workout started. Think I just need to get through this **rough patch** and get my legs to feel better. I know I'm in better shape than this.
0	Warmup (2 miles), 120-100-80-80-60 with 3:30 rest, Cooldown (2 miles)

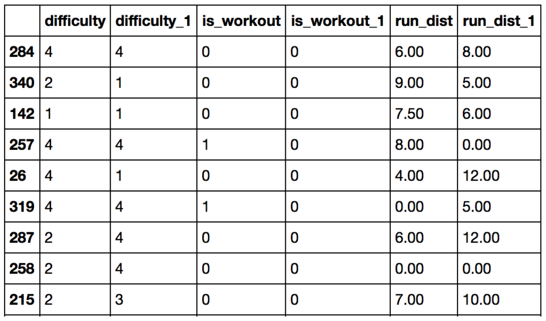
Using the same 407 notes, with manually labeled data for difficulty, we trained a multinomial naive Bayes model for this data. Getting exact accuracy was more difficult given the larger number of classes and fewer words indicating a specific class, but looking at the percentage of labels that were within certain ranges of the true label proved very helpful in evaluating our model. On average 0.42 of the difficulty labels were classified correctly, while 0.77 of the difficulty labels were classified within 1 of the labeled data. Looking at the closeness of difficulty scores is a good metric for accuracy, because the difference between two numbers (1 vs. 2, 2 vs. 3, etc.) is subjective and captures much of the same sentiment. Because the classifications of data with no sentiment, namely a 0 difficulty score, likely decrease the accuracy of the model without actually adding any necessary category, we also looked at the accuracies of the model when all observations labeled as 0 or predicted to be 0 are removed, as predicting a true 1 as a 0 is equally as incorrect as predicting a true 5 to be a 0. However, these two mistakes in predictions would adversely affect the accuracy of the model in varying degrees despite being equally incorrect. As expected, the accuracy metric of the model increased without zeros, as can be seen below in Table 2.

Table 2: Accuracies for Difficulty Naive Bayes

Accuracy	With zeros	Without zeros
Exact	0.421	0.500
Within 1	0.765	0.841
Within 2	0.873	0.943
Within 3	0.922	0.989

We were then able to use these models to take in all of the 1972 text observations in the data set and output subsequent workout classification and difficulty score instead of manually doing so. In the resulting data set, "run_dist," "is_workout," and "difficulty" are metrics for the current day and variables ending in "_1" are the same metrics for the next day. A snapshot of the resulting data set is below:

Figure 4: kNN Transformed Data



	difficulty	difficulty_1	is_workout	is_workout_1	run_dist	run_dist_1
284	4	4	0	0	6.00	8.00
340	2	1	0	0	9.00	5.00
142	1	1	0	0	7.50	6.00
257	4	4	1	0	8.00	0.00
26	4	1	0	0	4.00	12.00
319	4	4	1	0	0.00	5.00
287	2	4	0	0	6.00	12.00
258	2	4	0	0	0.00	0.00
215	2	3	0	0	7.00	10.00

7 k-Nearest Neighbors

Using our naive Bayes model, we were able to get a data set suitable for *k*NN classification. For the *k*NN classifier, we wanted to use the three metrics used to describe a day to prescribe the mileage and whether or not to do a workout for the next day. We omitted a difficulty prescription because, as is consistent with our definition of difficulty, it would not make sense to say "Tomorrow your difficulty should be 5" (Tomorrow you should feel really bad while you are running).

The kNN classifier works as follows: each observation is plotted in an n-dimensional space, where n is the number of predictors (in our case, 3). The distance between each point is calculated, generally using Euclidean distance. We specify a k value in our model, with k being the number of "nearest" neighbors we will look at. The optimal value of k is found through trial an error and two graphs of varying accuracies with different values of K can be seen in Figures 5 and 6. When we input a set of x-variables, the resulting prediction is the average of the y-variables of the k "nearest neighbors"—hence the name of the model. Options can be set to use a weighted average instead, valuing the "opinion" of neighbors in the set of k that are closer than others. The big assumption this model makes is that similar x-variable combinations have similar y-variables. In this case, it means that we think that days with similar mileages, difficulties, and workout classification will have similar next days of training.

7.1 Recommending Distance Run

We used the same x data to predict both running distance and workout classification. We evaluated our running distance model on R^2 value—how much of the predictive power in the model was due to our x-variables. For this, we trained models based on both "weighted" vs. "unweighted" distances and the number of neighbors k.

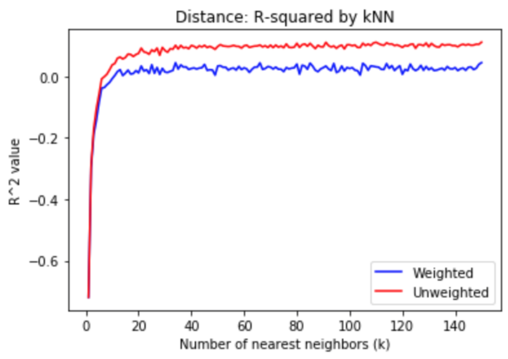


Figure 5: R^2 values by k

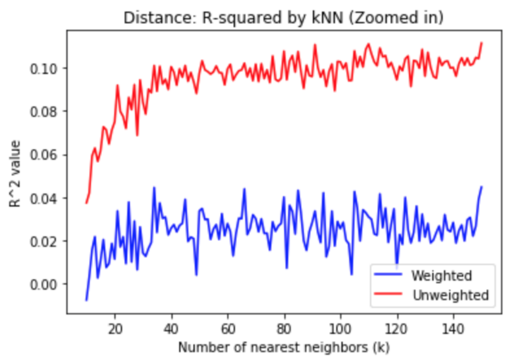
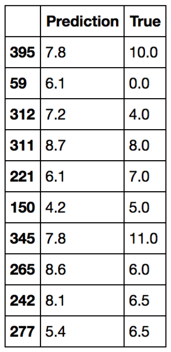


Figure 6: R^2 values by k (zoomed in)

The best model for running distance used the unweighted distances. This is probably due to the sometimes randomness of training and the subsequent randomness of the data. Just because a point is closer to another point, it does not necessarily mean that that point is significantly more similar than a point that is slightly further away, so weighting the distances actually decreased accuracy. We saw that R^2 tended to increase as k increased, but ultimately plateaued. If we continued to increase k, we would approach the mean of all of the y-values in the dataset and the model would get worse. For model simplicity, while maintaining high R^2, we chose our optimal k to be 25, meaning that the next day of mileage is determined as the average of the next days of the 25 nearest points to it.

While the R^2 value of the model is not very interpretable in regards to our data, we carefully looked over the predictions and true distances run to make sure our model seemed reasonable.

Figure 7: kNN Running Distance Truth vs. Predictions



	Prediction	True
395	7.8	10.0
59	6.1	0.0
312	7.2	4.0
311	8.7	8.0
221	6.1	7.0
150	4.2	5.0
345	7.8	11.0
265	8.6	6.0
242	8.1	6.5
277	5.4	6.5

These results are imperfect but very reasonable. Higher true values tend to have higher predictions and vice-versa. As seen in the second row of Figure 7, the prediction was very far off because the true distance run was 0. This may help explain the relatively low R^2 value of the model. Runners generally take one day completely off every week or two, and our model is unable to account for when these off days may appear.

7.2 Recommending Workout

We repeated this same process for workouts, using accuracy as our metric with which to evaluate the model, as it makes more sense for binary data. We trained both the weighted vs. unweighted distances as we did earlier, and also tested for different values of k to find the optimal k, finding that our best model was an unweighted kNN with $k = 10$ which yielded an average accuracy of 0.786.

Figure 8: Workout: kNN Accuracy

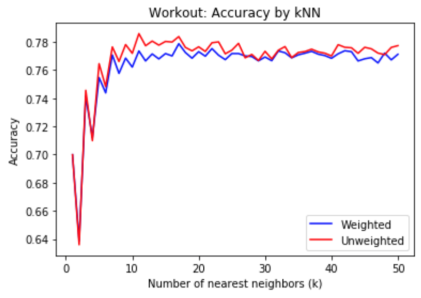
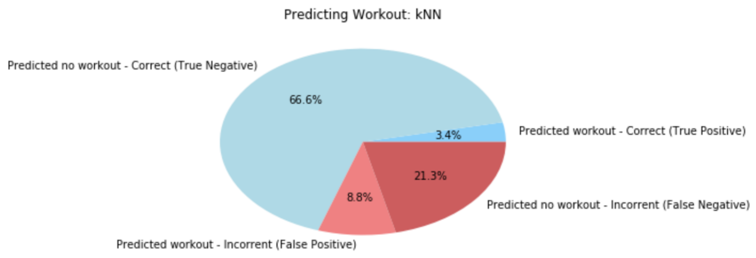


Figure 9: Workout: kNN Classification



This model likely did worse because workouts are much more dependent on the previous few days than just the previous day. Athletes typically do 2 or 3 workouts a week, and there are often 2 days of non-workouts in between workouts, so taking into account the last 2 or 3 days of training would greatly improve this model. As evidenced by the pie chart, 0.88 of all predictions are "no workout." Our model is very bad at predicting a workout the next day when there actually was one the next day. Given a workout the next day, we only predicted it as such 0.14 of the time. Our model had a relatively high accuracy because "no workout" is oversampled in the data set (0.778 of the data set is "no workout"), so simply predicting "no workout" each time would give us an almost equally good accuracy as our actual model.

Another thought here would be to use a logistic regression to recommend whether the next day should be a workout. Logistic regressions can be used to classify binary data, so this would be a reasonable model to use and could yield better results.

8 Results

To summarize the results above, our naive Bayes model for workouts averaged an accuracy of 0.926. Our naive Bayes model for classifying difficulty scores correctly classified within 1 of our manually labeled difficulty score 0.841 of the time.

Using kNN, we achieved an average R^2 value of 0.087 for our running distance recommender model. While the R^2 is not necessarily very interpretable here, the subsequent recommendations made by the system were very reasonable. Our kNN for recommending workouts the next day achieved an average accuracy of 0.786.

9 Ethical Discussion

Our ethical discussion will be composed of two distinct sections: the ethical questions related to (1) our specific recommender system, RoboCoach and (2) an advanced, optimal version of RoboCoach. The following two sections will be organized according to the ethical questions we considered within that section, our conclusions for the specific questions, and also a thorough ethical discussion that led to our conclusions.

RoboCoach
In this section we will consider the following questions: Was it ethical to (1) use peoples sensitive data? (2) create this recommender system that only works well for a specific, already privileged demographic, namely competitive 18-22 year old, male runners? and (3) create a system with potential inaccuracies that the users do not know about?

First we considered the ethical implications of using sensitive data as training data for our naive bayes classifiers as well as our K-nearest neighbors algorithm. The data that we used include potentially personal data, given that the running logs usually contained subjective feelings about how a days workout went. We concluded that it would have been ethically impermissible to use this data if any of the following conditions were not met: (1) the explicit permission to use the running logs from each individual who provided us with their running logs, (2) the full knowledge of what exactly the running logs would be used for, (3) the use of the running logs only for the use cases that were agreed upon, and (4) the anonymity of all the data outside of the algorithm. Not only did we receive explicit verbal and written permission to use the running logs for our research, we received the NIH PHRP training for using human subjects in our research (certificate is attached on canvas along with project). Moreover, we only used the running logs for the use cases agreed upon in the written consent and we also made sure to keep the data entirely anonymous so that the users cannot read the logs in any way. They were simply used to train our algorithms. Given that our project fulfilled the four criteria laid out above, we conclude that it was ethically permissible to use potentially sensitive data in the creation of our recommender system.

The second ethical question we consider is the permissibility of creating a recommender system catered only towards a specific demographic. The data used to train our recommender system came from a biased group of individuals, namely 18-22 year old, division one, male runners. Consequently, the recommender system will only work well for this specific demographic and would not provide great recommendations for any other types of runners. For example, our recommender system may return a recommendation to run 14 miles and a workout the day after running 3 miles and no workout. While this makes perfect sense for a competitive college athlete, this sort of recommendation would be a bad recommendation for other demographics of people. Ultimately, we conclude that it is ethically permissible to create a recommender system that works only for a specific demographic as long as the following conditions are met: (1) the recommender system is labeled only for the use of this specific demographic so that it is not misused and (2) the specific system does not create a disparate impact that helps this one group advance in a humanly important way that other groups cannot.

The first condition is met as we are properly identifying this bias and only condoning the use of such a system within the bias. The second condition is met because this recommender system does not provide an additional humanly important advantage to the specific demographic for which it is created. This is clear if we examine the specific demographic for which this recommender system was created. Almost all people who fit into the category of male, 18-22 year old, division one cross country runners likely already have access to a coach. Moreover, the few that do not have access to a coach likely have been running long enough that they know enough about cross country training that each could create their own plans of equal caliber to our recommender system anyways. Subsequently, this recommender system does not create additional opportunities in its current form so does not create a disparate impact between the target demographic and other demographics.

Finally, we considered the the ethical permissibility of creating a system that has potential inaccuracies that could influence the user negatively without their knowledge. In order to fully tease out this concern, we will (1) provide an example of an inaccuracy in our system that could negatively impact the user, (2) consider the ethical implications of this inaccuracy, and (3) conclude whether or not it is ethically permissible to create a system with these inaccuracies.

Although our system usually provides accurate recommendations, nearly [insert data here], sometimes the naive bayes classifiers still completely misclassify the text by classifying a day as much harder or much easier than it actually was. Both cases are problematic, as the first could lead to injury and the second would lead to less potential improvement of the runner. Given the gravity of the first, we will focus on an example of a mistake of the first kind in our data. For example, one anonymous entry in a running log is written as follows: "Day off, feeling a bit sick (don't do too well with travel), so day off was good." Given that the runner did not feel good this day and actually felt sick, we labeled the difficulty of this day as a 5, the most difficult, so that the next days would include more rest until the runner is better. However, our classifier labeled this day as a 1, likely because of the keywords day off, good, and well. The classifier had difficulty picking up real sentiment of the log, but instead picked up on the optimistic, yet not representative, second part of the text. This sort of misclassification is problematic because the user does not necessarily know how the naive bayes classifier works and so may put too much trust in the system and follow this recommendation beyond better judgment. Although our classifier correctly predicts a difficulty score within 2 of the correct difficulty score 90% of the time [include link or reference], there is still a 10% chance that the classified difficulty scores are three or four away from the true values. A mistake of this sort could potentially lead a runner to injury, especially because the runner does not know exactly why the system offered the recommendation it did, and so could naively trust the system too much.

Given that this sort of misclassification occurs a nontrivial amount of time, we must consider the ethical permissibility of using this system that could potentially injure the user. We propose that the use of this system is permissible if (1) the risks of using such a system are clearly outlined and (2) it is clear that user should use their better judgment and use the system within its ability, namely as a recommendation rather than a strict guideline for what to do. In other words, the user must be fully aware of the abilities and the subsequent inabilities of the system.

Optimal RoboCoach

For the optimal RoboCoach section, we will consider the ethical permissibility of these two outcomes: (1) the system may take away from the artistic value of the sport of Track and Field, and (2) an optimal system may optimize too far and potentially cause adverse health effects.

First, we will consider question of automation in art and sport. Specifically for this case, does the added automation of the coaching position in long distance running take away from an essential piece of the sport? For example, the roles of officials and referees in different sports such as basketball or baseball could likely be automated such that results of rule infringements would be more accurately captured. However, sports leagues chose not to automate such roles because of the importance of the human aspect of each role to the sport. If we consider the purpose of sport in the context of society, it seems clear that the artistic value of the sport must be maintained at the cost of accuracy. Part of the beauty of sports is the humanity that exists within the sport, namely the human struggle, hard work, and cooperation in order to achieve a common goal. For a similar reason, performance enhancing drugs are not allowed in many sports leagues; these take away from a level playing field and ultimately take away from the value of the sport. Even though athletes across the board might improve their game by taking performance enhancing drugs, this sort of enhancement takes away from the perceived humanity of the games.

Ultimately, we would like to argue that the relationship between a coach and an athlete within the context of Cross Country and Track and Field is incredibly important such that the automation of this aspect of the sport would take away from the true essence of the sport itself. The coach and athlete typically create a strong bond throughout years of training and communicate with each other on a deeper level to understand each others needs at the time, often even outside of the physical training itself. Moreover, the coach often adjusts workouts and distances depending on how the athlete feels at a given moment. Despite this conclusion, we would like to argue that from an ethical standpoint, automation of coaches is ethically permissible because of three reasons. First, the question of art and sport is not one of ethical significance; this question is best answered outside of an ethical context. Second, access to an automated coach, even an optimally automated coach, will still lack some of the human to human connection that exists between an athlete and a coach. This in turn means that access to an automated coach may not actually help the athlete in a way that creates a disadvantage to those who do have access to a coach. Finally, the automation of coaching may actually close the gap between those who simply do not have access to a coach and those who do rather than widening the gap of inequality.

The second question will consider is the ethical permissibility of creating an optimal system that could have potentially adverse health effects on users. For example, a runner may want to optimize for performance to the extent that she will not maximize her own health. This case is actually extremely common, as the most competitive athletes in most sports are actually not as healthy as they can be because they optimize too specifically for their sport. Consider the case of long distance running. A runner will be more competitive if she runs 80 miles a week over 40 miles a week, but is at higher risk of injury and bone/tendon deterioration. In the sport of Cross Country, there is a fine line between optimizing for performance and simply maintaining health. Often, the runner pushes to optimize for performance but the coach must hold the runner back to the extent that she does not get injured or the coach pushes to optimize for performance but the runner knows her body well enough to foresee injury. Either way, the checks and balances that exist between runner and coach often help the athlete in the long run and must be considered in the case of the recommender system. The ethical difficulty arises because even an optimal recommender system will likely just provide the user with the information the user needs. In other words, if the user wants to optimize for performance and decrease the likelihood of good health, then should the system listen to the needs of the user or override the system and provide a different recommendation that maintains health?

We argue that the creation of an optimal system that makes decisions that affect a persons health is ethically permissible as long as the following two conditions are met: (1) the system warns the user about the potentially adverse health effects of optimizing for performance to an unhealthy degree thereby ensuring a check to the athlete, and (2) the system learns over time how the athlete responds to certain types of training and subsequently learns the optimal training load that maximizes both health and performance. Ultimately, if these two conditions are met, then the system could actually provide an even better check than a coach would because of its impartial and objective nature. Each decision will be based off of sound data and evidence rather than speculation about ones health based on incomplete evidence.

10 Conclusions

In conclusion, we were successfully in our technical implementation of a recommender system that provides automated training suggestions based on a previous day's training. The results were sound and reasonable enough that a runner could actually use this system for a general training guideline in the future. However, there were some places where our models, both naive Bayes models and also both k-nearest neighbors models, could be improved. A few of these potential improvements were discussed throughout the results of our project, and included examples such as using a logistic regression to recommend whether the next day should be a workout or removing any sorts of days without any sentiment attached for the difficulty scores. With regards to the ethical component of the project, we concluded that overall, the creation of this system in the way that it was trained and implemented, is ethically permissible given that a few different conditions are met. Similarly, we argued that, for the most part, a perfectly optimal recommender system of would be ethically permissible to create with regards to its effect on the sport because this sort of system would not create a disparate impact even though this sort of system could take away from the artistic value of the sport. However, we also determined that the latter consequence was not an necessarily ethical consequence but rather a question of pragmatics. Again, we ultimately argued for the ethical permissibility of such a system given a few different conditions are met in the use and distribution of such a system.

11 List of References

Given the separate technical and ethical components of our project, the references for our project are split up into two distinct parts: one for the technical and one for the ethical.

Technical:
We used the first three references to help us understand and implement the naive Bayes classifier in the first part of our technical implementation. The first reference helped provide a conceptual understanding, the second pointed us to further resources, and the third is the python package we used to run the algorithm. We used the two references in order to help us understand and implement the second part of the technical implementation. The fourth helped provide a conceptual understanding whereas the last is a link to the documentation for the python package we used to run the K-nearest neighbors algorithm.

1. Scott Kuindersma. Computer Science 182: Artificial Intelligence. Lecture 18: Classification and Regression. 2017.

2. Sebastian Gehrmann. Personal interview. Grad student offices in Maxwell Dworkin. November 2017.

3. 1.9 Naive Bayes. Scikit-Learn Documentation,
http://scikit-learn.org/stable/modules/naive_bayes.html

4. Scott Kuindersma. Computer Science 182: Artificial Intelligence. CS182 Lecture 19: Clustering. 2017.

5. 2.3 Clustering. Scikit-Learn Documentation,
http://scikit-learn.org/stable/modules/clustering.html#clustering

Ethical:
Most of the ethical references we used for our project were taken from the required or optional list of readings provided in class. Each reference helped with the overall examination of the ethical questions involved in our project.

1. V. Goel. As Data Overflows Online, Researchers Grapple With Ethics. New York Times, August 12, 2014.

2. J.A. Konstan and J.T. Riedl. Recommender Systems: from algorithms to user experience. User Modeling and User-Adapted Interaction (2012) 22:101-123.

3. S. Barocas. Data mining and the discourse on discrimination. Data Ethics Workshop, Conference on Knowledge Discovery and Data Mining. 2014.

A Appendix 1: System Description

In order to run our system and generate the output we discussed in the write-up, simply go to the github repository here: https://github.com/dalpert/RoboCoach. In this repository, you will find a a file named CS108 naive Bayes.ipynb. If you open this jupyter notebook and run the code step by step, each of the pieces of data with regards to the naive Bayes implementation that were referenced in the project will be generated. Similarly, if you open the CS108 kNN.ipynb file and run each piece of code step by step, then the output discussed in the write-up with regards to kNN will be generated.

B Appendix 2: Team Member Contributions

Sam and Daniel worked together on each part of the project and put in an even amount of work into the project. In terms of the technical portion, Daniel and Sam both worked together in labeling data and writing code for the naive Bayes classifier as well as the K-nearest neighbors algorithm. In terms of the ethical portion, both Daniel and Sam worked together to consider each potentially ethically significant question that arose from the project and come to a conclusion about the ethical nature of each situation.

Word Count: 6592