Predict the Winner: Three-Year Model

Pulkit Khandelwal* (260717316)
Monica O Patel* (260728093)
Gary Corcoran* (260681354)
*McGill University

Introduction:

Classification is used to categorize data and regression is used to estimate relationship between variables to predict output on a set of input data. We employ classification techniques such as Logistic Regression, Naive Bayes, Linear Discriminant Analysis, and Quadratic Discriminant Analysis along with regression techniques such as Linear Regression for the following problem.

Comprehensive analysis has been given on the training, validating and testing methodologies. Results follow some rich discussion. Variations in models and comparison with other models has been given due importance.

Problem Statement:

Given the data of 8711 runners for events over the last 4 years, we are given the task to predict the results for the upcoming 2016 Montreal Marathon.

1. Predict which participants (who have been part of some race event in the previous years) will come to the 2016 Montreal Marathon on 25th September 2016.
2. Assuming that all participants come for the marathon, their race completion times have to be estimated.

Feature Representation:

A set of fourteen features have been selected because they try to capture the essence of the extracted raw data with the three specific features specifying the correlation between the past years and the current year 2016. Some domain specific knowledge should be incorporated to achieve the best set of features.

Table 1 through 4 depicts the features taken into account for training and testing phases Note: In the tables: 12 denote 2012; 13 denotes 2013 and 14 denotes 2014

TASK Y1:
NAÏVE BAYES
Training Method:

In machine learning, Naive Bayes is a probabilistic classifier that has strong independence assumption between the features. We use this independence assumption to predict which previous runners will participate in the upcoming 2016 Montreal Marathon.

Design and distribution choices:

Fourteen features have been selected for the training method. Four out of these ten features have Boolean values. The rest of the ten features have continuous input values. One needs to carefully select the distribution function for the likelihood. We consider two distribution scenarios for this task:

[A]. A univariate Gaussian distribution for the ten continuous features and a Bernoulli distribution for the four Boolean feature set. One can include such distributions because Naive Bayes have a strong assumption on independence between the features.

[B]. A univariate Gaussian distribution for all the features in the input space.

Posterior probability: The first ten features assume Gaussian and the remaining four assume a Bernoulli distribution. The equation below shows this form.

$$P(y=k|x) = \frac{\prod_{i=1}^{10} P(x_i|y=k)\ \prod_{i=11}^{14} P(x_i|y=k)}{P(x)}$$

During testing phase, data from years 2013 through 2015 acts as the input and we predict the outcome for 2016 for both classification and regression tasks.

Training:
Case A

1. Divide the dataset into 5-folds. 80% of the data is kept as training data and 20% as validation data.

2. Calculate mean and standard deviation for each of the continuous features across all the training examples. This mean and standard deviation will be used to estimate the likelihood probabilities of each of the input feature for the validation data.

3. Calculate the likelihood for the Boolean data and calculate the prior probabilities using the equation.

4. Now, on the validation set: use the Gaussian function to evaluate the likelihood for each feature for each test example using the mean and standard deviation as calculated in step 2. Also, evaluate the likelihood for each of the Boolean feature for each test example.

5. Calculate the posterior probability for each of the test example by multiplying with the appropriate prior.

6. Compare the P(Y=0|x) and P(Y=1|x) for each test sample:

If P(Y=0|x) > P(Y=1|x) then test example classified as Class 0
Else P(Y=1|x) > P(Y=0|x) then test example classified as Class 1

Case B

Implement similar aforementioned steps assuming all the features to be continuous.

Case C

The highly optimized scikit learn library has also been used separately to train and test the data using a Gaussian distribution for all the features. This acts as a good comparison benchmark for our implementation.

Prediction

Train over the entire dataset of 8711 participants. Predict on the prediction dataset as mentioned in the Feature Description section for all the three cases A, B and C.

RESULTS: Naïve Bayes

Calculate evaluation metrics: training and validation error; confusion matrix to estimate accuracy, precision, recall and F-1 score. Receiver and Operating Curve, Area under the curve have also been plotted. These graphs are shown in the appendix.

As apparent from Table 5, assuming a Gaussian distribution for the entire feature space gives better performance than the dual distribution case of Gaussian and Bernoulli. Also, the implementation from first principles give exactly same result as obtained from the case of the Gaussian distribution.

The dataset set has been split in a ratio of 1:5 for training and validation set. Cross validation acts as a sanity check. It utilizes entire dataset for training and testing (here, validating) the learned model. This way one can be assured of how well the model works on future unseen data and avoid the problem of overfitting the data by working as a good generalizer. Hyper parameters can also be fine-tuned using cross validation.

See Appendix 1 for the ROC Curves for all the implementations. ROC helps to estimate optimal models amongst a set of models. The area under the curve for Case A and C are same and greater than Case B which depicts that case B does not provide us with a good model as the accuracy is less and misclassification rate is high. Additionally LDA and QDA comparison denotes that these models give nearby results to Class A and C

Prediction

The above Naive Bayes learner is used to predict which participants will participate to the 2016 Montreal Marathon. Out of 8711 previous participants, 1961 have been predicted to race in the 2016 Montreal Marathon. The reported participants have been predicted to participate by the model implemented from first principles (case A: Gaussian). See the output in the accompanied prediction.csv file.

Surprisingly! The same number of participants and the same participants have been predicted to participate by the model implemented from first principles (case A: Gaussian) and from using the scikit learn library (case C).

Discussion on Y1: Naive Bayes Classifier:

1. Why is Naive Bayes Classifier good even though it has a strong independence assumption?

For most practical tasks Naive Bayes is considered “good enough” and oftentimes used a baseline classifier. It is easily implemented and offers a quick solution.

2. Disadvantages of Naive Bayes?

Naive Bayes assumes a simple model and hence complex models do not fit well.

3. Why these features for Naive Bayes?

First of all there is no reason at all to choose one classifier over another based on evaluation metrics alone. Oftentimes feature selection and engineering, along with other hyper parameters and optimization techniques play a crucial role to select a given classifier. That said, if one choose to go with a particular classifier, he/she should be very careful to apply the same classifier to some other task (where features might be different) as chances are that the chosen classifier might fail miserably.

Due to the design of the feature space, one can see that the matrix is sparse. Should that affect the solution? No, because the independence assumption takes care of that. Also, due to such a large number of examples and appropriate number of features Naive Bayes gives good accuracy and prediction results.

A sparse matrix might affect the result of a prediction but a regularizer in logistic regression will squeeze the dimensionality and help us escape its curse. As for Naive Bayes, it doesn’t matter much as evident from the results due to its independence assumption. There no matrix inverses or multiplications to be computed. Only probabilities have to be computed!

4. Combination of different distributions?

For continuous data one should use a probability density function such as a multinomial Gaussian function. For Boolean/binary data, probability mass function is used such as a Bernoulli distribution. One can also choose distributions according to the types of feature. Here, we observe that choosing a Gaussian distribution gives better results than the mixture of models!

Note: One can also use log probabilities to avoid the problem of errors due to floating point numbers. This is also helpful when the probabilities are small and one does not want the product of probabilities to vanish to zero!

5. When does Naive Bayes fail?

A quote from a paper by Pedro Domingos summarizes this well:

"Although the Bayesian classifier's probability estimates are only optimal under quadratic loss if the independence assumption holds, the classifier itself can be optimal under zero-one loss (misclassification rate) even when this assumption is violated by a wide margin."

TASK Y1:
LOGISTIC REGRESSION

Logistic regression is linear model of classification. It finds a best fitting model to describe the relationship between response and independent features. We use this model for predicting if a runner will be participating in upcoming 2016 Marathon Oasis event.

Design and distribution choices:

For this model learning rate is 0.001 with number of iterations = 2500 which are chosen by optimizing on running time and accuracy obtained which can be seen in following analysis:

We plotted graphs of Cost function v/s parameter for various values of learning rate and no of iteration and checked for what values of alpha and iteration the cost function converges as well as we get good accuracy. Check Appendix 2 for the graphical analysis

Throughout the gradient descent cost function value should decrease. So learning rate should be small enough so that the cost function value does not diverge and at the same time should be large enough that the parameters are learned with less number of iterations.

We can see from and table 7 that the cost function was diverging for greater weights; and accuracy decreased for small theta given same number of iteration. Therefore we chose theta to be 0.01 and no of iteration = 2000

K Fold validation:

Model was cross validated using K fold validation to avoid bias and limit the problem of over fitting. We used 4-Fold validation method over the training data:

Discussion on Y1: Logistic Regression Classifier

1. Accuracy of the fit is poor with smaller learning rate and less number of iterations because cost function has still not converged.

2. If learning rate is increased, local minima is possibly reached in fewer steps but there is also possibility of crossing over the local minimum and attaining a higher value instead. Learning rate should be small enough such that cost function does not diverge.

TASK Y2:
LINEAR REGRESSION
Design and distribution choices:

The features that are used for the linear regression portion of this algorithm differ slightly than the classification problem. For the linear regression algorithm the features that are used include Age, Gender, Average Speed, and Rock N' Roll marathon duration (time required for participant to finish marathon). The age and gender features are encoded as per the classification task. The average speed features, however, are divided into 15 sub-features. They consist of the average speed for the past three years for each of the following categories: 1km, 5km, 10km, 21km, and 42km. It was decided to divide these features in this respect as it better demonstrates trends compared to averaging speeds over all categories. One example of this is if someone only runs 1km or 5km races they will unlikely have a fast time for the 42km race, however if the participant ran a 42km race in a fast time in the previous year they are likely to yield a fast time again. Similar to the average speed features, the Rock N' Roll durations were divided into their last 3 years; however, these features only include the duration of the full marathon (i.e. 42km).

Initially, we decided to utilize the closed-form solution to the least-squares problem using the features as they are described above. However, to obtain a better understanding of the linear regression algorithm, additional approaches were taken. Our second approach added a regularization term to the closed-form solution. The regularization parameter was varied on a log scale of (0.001, 0.01, 0.1, 1, 10, 100). Using these two methods, we also tested encoding the features using a polynomial basis. We utilized the original features plus seven additional features. These features included age * gender, five features for each category of race multiplied by each year (e.g. avg. speed [1km, 2012] * avg. speed [1km, 2013] * avg. speed [1km, 2014]), and lastly the duration of each Rock N' Roll race multiplied by each other for each year (i.e. Rock N' Roll time [2012] * Rock N' Roll time [2013] * Rock N' Roll time [2014]). It is worth noting that this is not exactly how the polynomial basis are defined, however, we decided to only include the polynomial terms we thought would provide valuable information to the learning algorithm. Using these polynomial bases, the closed-form solution (with and without regularization) was tested. The last approach that was tested was utilizing gradient-descent. The learning rate for this algorithm was varied from (0.001, 0.01, 0.1, 1, 10, 100).

RESULTS: Linear Regression

The following tables demonstrate the validation and training error received for each linear regression algorithm we tested. Table 4 and 5 use the SkLearn library and thus are not implemented by us; they are used for comparison. The error is calculated by the mean squared error from the true value (in hours). We used the 2015 Rock N' Roll marathon times as our true values and we attempted to predict these values using the three-year model. Standard 5-fold cross validation was used to calculate the validation and training errors.

Discussion on Y2: Linear Regression

Table 10 shows the closed-form linear regression algorithm as we vary the regularization parameter. As we increase the regularization parameter the training error continues to increase. This makes sense as we are now generalizing to the data. On the other hand, the validation error first decreases and then increases. This also looks accurate since at a small regularization parameter, we are not generalizing to the testing data enough and at large values we are generalizing too much. A regularization parameter of 0.001 yielded our best validation results at 2.3646 (MSE).

Table 11 shows the same closed-form linear regression algorithm using a polynomial basis. We only selected a small portion of the complete polynomial basis to work with. As one can see from the table, the regularization parameter effects the validation and training errors similar to the regular basis functions, however, the error is slighting larger. I believe this is due to the fact that the polynomial features we included did not add that much information to the algorithm and mainly just added more noise.

Table 12 shows the results of using gradient-descent. At small learning rates, the algorithm looks to perform the best. At larger learning rates, the algorithm has a hard time converging and will oscillate around the minimum and in most cases begin to increase. We see this trend as our learning rate increases past 0.01. The algorithm does not perform as well as the closed-form solution since we do not incorporate a regularization term.

Table 13 and 14 demonstrate the SkLearn libraries implementation of ridge regression and gradient-descent. These results are provided merely for comparison to our algorithms. As one can see our closed-form solution provides better results than the SkLearn ridge regression (although I cannot say why) and our gradient-descent performs worse than the SkLearn implementation. This is mainly due to the fact that our gradient-descent algorithm is very basic and does not include a lot of optimization.

Unfortunately, our best algorithm (the closed-form regression with regularization) still performs poorly (2.3646 MSE in hours). We believe the reason behind this is not having enough features. Initially we wanted to keep our feature dimensions small in order to avoid overfitting on the training data. However, once we started producing results, we noticed that we were receiving large errors. When we looked at our input feature matrix we noticed that a lot of elements were zero, and thus, the data did not contain as much information as we originally thought. Thus, our largest limitation is not having enough features. For future iterations, we would like to improve the quality of our input feature matrix by including features that will not yield a sparse matrix.

Tables and Graphs

Table 1. Task Y1: Classification Training Data

Age	No of Race Completed			No of Race Participated			Average Speed			Gender	Participated in Rock N Roll			Y: Participated in Rock N Roll
	2012	2013	2014	2012	2013	2014	2012	2013	2014		2012	2013	2014	2015

Table 2. Task Y1: Classification Testing Data

Age	No of Race Completed			No of Race Participated			Average Speed			Gender	Participated in Rock N Roll		
	13	14	15	13	14	15	13	14	15		13	14	15

Table 3. Task Y2: Regression Training Data

Age	Gender	Average Speed															Rock N Roll run time			Rock N Roll Time (Y)
		2012					2013					2014					12	13	14	15
		42 km	Half Marathon	10 km	5km	1km	42 km	Half Marathon	10 km	5 km	1 km	42 km	Half Marathon	10 km	5 km	1 km				

Table 4. Task Y2: Regression Testing Data

Age	Gender	Average Speed															Rock N Roll run time		
		2013					2014					2015					13	14	15
		42 km	Half Marathon	10 km	5 km	1 km	42 km	Half Marathon	10 km	5 km	1 km	42 km	Half Marathon	10 km	5 km	1 km			

Table 5. Evaluation Metrics: Comparison between the different distribution and implementations

Distribution/ Implementation	TP	TN	FP	FN	Accuracy (%)	Precision(%)	Recall (%)	F-1 Score (%)
A: Gaussian	2416	4737	1222	336	82.11	85	82	83
B: Gaussian_Bernoulli	2356	3496	2463	396	67.17	77	67	68
C: Sci-kit_gaussian	2416	4737	1222	336	82.11	85	82	83
LDA	2515	4717	1242	237	83.02	86	83	84
QDA	2424	4812	1147	328	83.06	85	83	84

where TP: true positive; TN: true negative; FP: false positive; FN: false negative

Table 6. Evaluation Metrics: Error rates for 5-fold training and cross validation sets

Fold Number	Accuracy_train (%)	Accuracy_validation(%)	Error_train(%)	Error_validation(%)
1	82.19	81.27	17.81	18.73
2	82.35	81.20	17.65	18.80
3	81.79	83.06	18.21	16.94
4	82.19	82.26	17.81	17.74
5	81.99	68.42	18.01	31.58

Table 7. Learning Rate and Iterations: Logistic Regression

Learning Rate	# Iterations	Accuracy(in %)	True Positive	True Negative	False Positive	False Negative
0.1	300	80.81	2386	4654	1305	366
0.01	300	79.80	2244	4708	1251	508
0.001	300	37.26	2682	564	5395	70
0.001	600	48.14	2550	1644	4315	202
0.001	1200	71.79	2382	3872	2087	370
0.001	2000	75.47	2256	4319	1640	496

Table 8. Logistic Regression K fold Training

Fold Number	Accuracy	True Positive	True Negative	False Positive	False Negative
1	68.09	7	4739	15	2209
2	68.05	8	4735	19	2207
3	68.11	8	4738	15	2207
4	68.11	9	4737	16	2206

Table 9. Logistic Regression K fold Testing

Fold Number	Accuracy	True Positive	True Negative	False Positive	False Negative
1	68.81	2	1196	8	535
2	68.71	0	1197	5	540
3	68.55	4	1191	5	543
4	67.42	1	972	2	468

Table 10 - Closed-Form Regular Basis Functions: Linear Regression

Regularization Param.	Error Validation	Error Training
0	2.3890	3.4060
0.001	2.3646	3.4241
0.01	2.4056	3.4592
0.1	2.6099	3.5417
1	2.6736	3.5309
10	2.8025	3.5357

Table 11 – Closed-Form Polynomial Basis Functions: Linear Regression

Regularization Param.	Error Validation	Error Training
0	3.2971	3.5229
0.001	3.1971	3.5227
0.01	3.2536	3.5380
0.1	3.3277	3.5434
1	3.4289	3.5591
10	3.5049	3.5605

Table 12 – Gradient Descent: Linear Regression

Learning Rate	Error Validation	Error Training
0.001	3.7048	3.7943
0.01	4.1722	3.2632
0.1	2307.4202	2321.9630
1	4231.6190	4322.5034
10	41776.6092	42579.7164

Table 13 – SkLearn Ridge Regression: Linear Regression

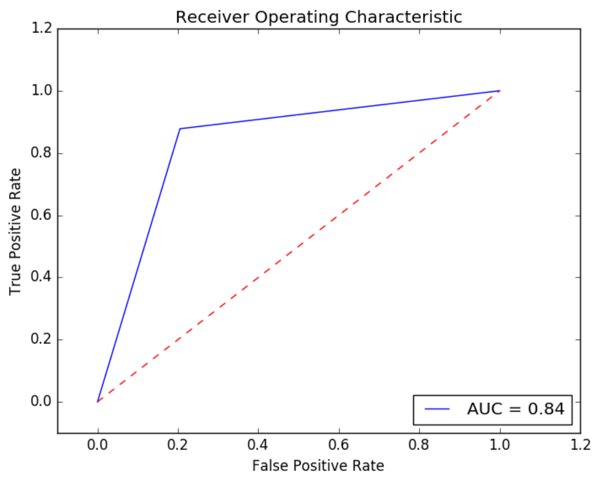
Regularization Param.	Error Validation	Error Training
0	9.5597	9.8427
0.001	9.5587	9.8427
0.01	9.5595	9.8356
0.1	9.5581	9.8435
1	9.9586	9.8505
10	9.5912	9.8894

Table 14 – SkLearn Gradient-Descent: Linear Regression

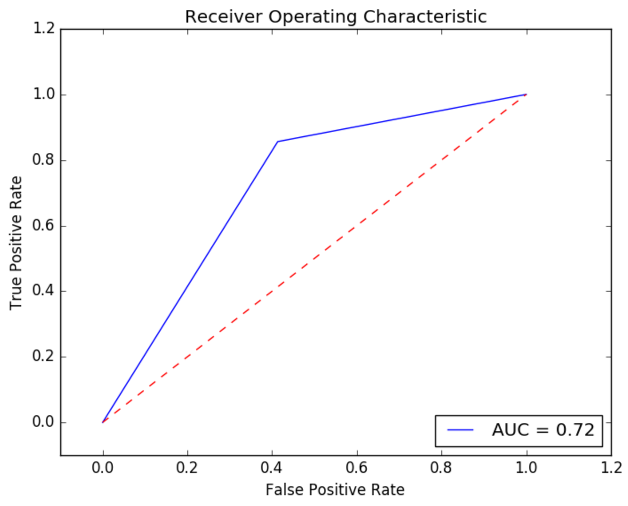
Learning Rate	Error Validation	Error Training
0.001	4.4781	4.7546
0.01	5.0164	5.3014
0.1	6.6113	6.7676
1	7.1381	7.1989
10	7.0893	7.1465

APPENDIX 1: ROC Curves for naive Bayes using five variants:

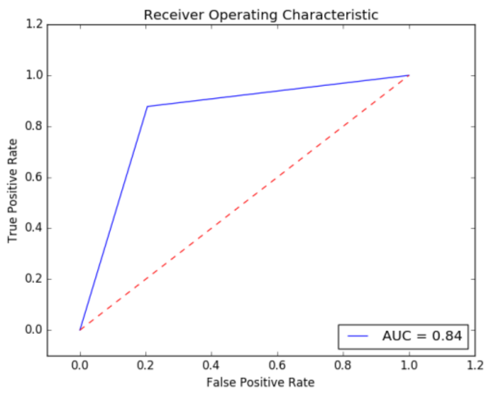
Graph 1. A: Gaussian



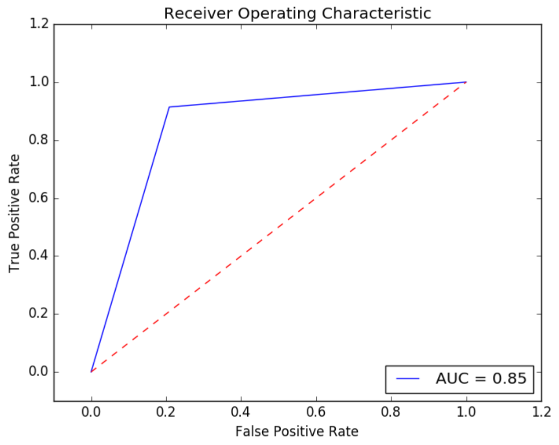
Graph 2. B: Gaussian_Bernoulli



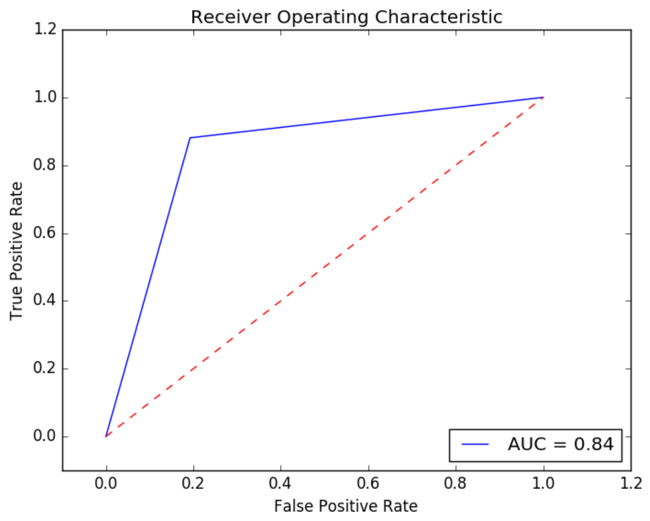
Graph 3. C: Sci-kit



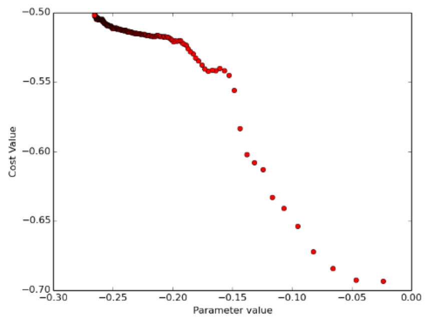
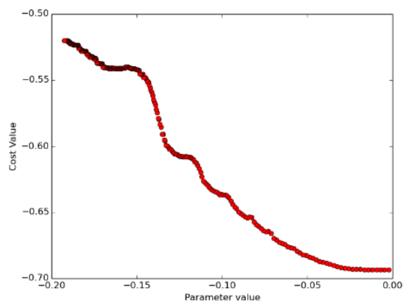
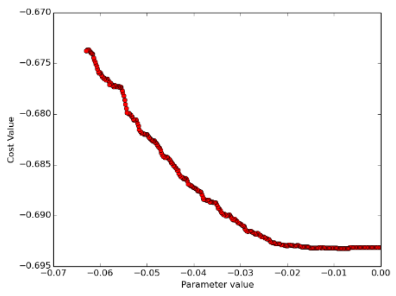
Graph 4. B: LDA



Graph 4. B: QDA

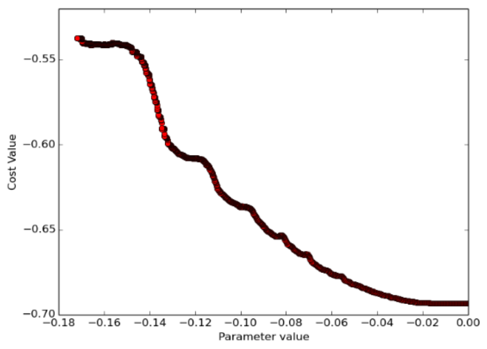


APPENDIX 2: Learning Curves and ROC for Logistic Regression
X- Axis Parameter Value
Y- Axis Cost function Value

Learning Rate = 0.1	Learning Rate = 0.01	Learning Rate = 0.001
		

Final Graph

Alpha = 0.01 Iteration = 2500


Cost Value
Parameter value