Statistical Learning for Esports Prediction

Kevin Bailey

April 30, 2020

League of Legends (LoL)

- One of the most popular video games, and is free to play.
- Matches consist of 10 players, 5 on each team (blue side vs. red side).
  - Each player has their own role, typically associated with a "lane".
  - Each player picks a unique champion (character) for the match that has its own unique set of abilities and starts each new match at level 1.

League of Legends (LoL)

- One of the most popular video games, and is free to play.
- Matches consist of 10 players, 5 on each team (blue side vs. red side).
  - Each player has their own role, typically associated with a "lane".
  - Each player picks a unique champion (character) for the match that has its own unique set of abilities and starts each new match at level 1.
- Champions earn gold by being the last one to hit a monster, resulting in that monster dying, by killing other champions, or taking objectives.
- Champions also earn experience by being around a death of a monster, champion, or objective, which levels them up to enhance their abilities and stats.
- The ultimate goal of each match is to reach the enemy's base and destroy their Nexus, which is protected by turrets.

The Map (Summoner's Rift)

Esports

- Short for *electronic sports*.
- A form of competitive sports using video games across many genres.

Esports

- Short for *electronic sports*.
- A form of competitive sports using video games across many genres.
- Popularity and monetary involvement are growing quickly, especially for LoL.
  - 2019 LoL World Championship peaked at 44 million viewers, being broadcast in 16 languages.
  - Average salary for professional LoL player increased from $105,000 in 2017 to $300,000 in 2019.
- Esports organizations have been hiring full staffs like traditional sports.

League of Legends Championship Series (LCS)

- The main circuit of professional LoL, which started in 2013.
- 10 teams in each region compete over a 9 week period.
- Winners of each region are guaranteed a spot at the World Championship.
- Broadcasts live each weekend to hundreds of thousands of viewers.

VIDEO: Teamfight from 2016 LCS match C9 vs TSM

Data

Pre-Transform

- Data came from Oracle's Elixir website: https://oracleselixir.com/match-data/
- Only consisted of professional LoL matches.
- Raw data consisted of $8,058$ rows with 90 columns.

Data

Pre-Transform

- Data came from Oracle's Elixir website: https://oracleselixir.com/match-data/
- Only consisted of professional LoL matches.
- Raw data consisted of $8,058$ rows with 90 columns.
  - Much of that data was unnecessary.
  - Every 12 rows corresponded to 1 match.
  - Difficult to split and work with.

Data

Pre-Transform

- Data came from Oracle's Elixir website: https://oracleselixir.com/match-data/
- Only consisted of professional LoL matches.
- Raw data consisted of $8,058$ rows with 90 columns.
  - Much of that data was unnecessary.
  - Every 12 rows corresponded to 1 match.
  - Difficult to split and work with.

Post-Transform

- Converted matches from 12 rows into one row with relevant predictors as columns.
- Transformed data had 671 rows and 29 columns.

Our response is the result of the match with respect to the blue side, coded as a 0 for a loss and a 1 for a win.

Data (Pre- and Post-Transform)

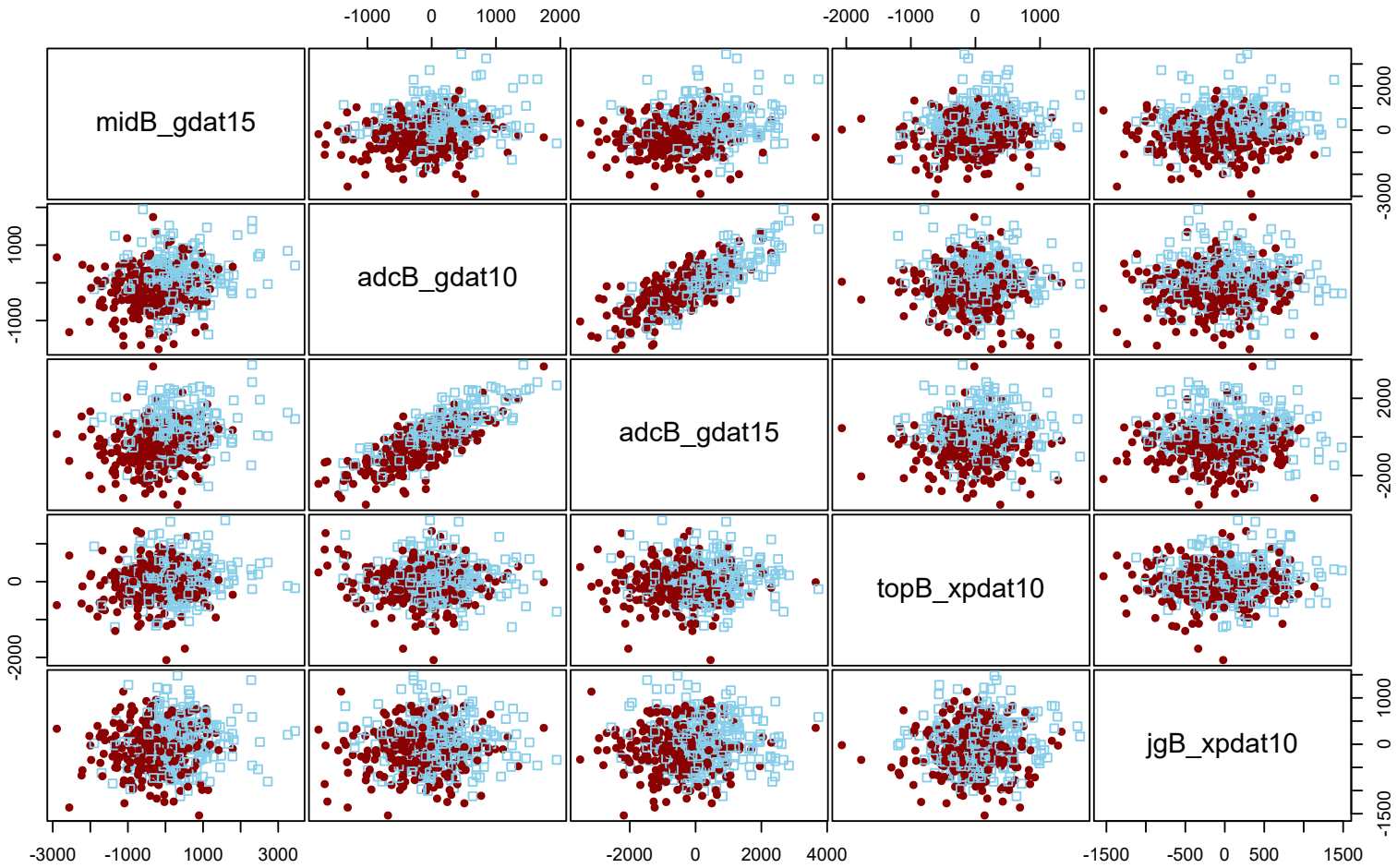
Side	Position	Player	Team	Gold at 10 Minutes
Blue	Top	Licorice	Cloud9	3668
Blue	Jungle	Svenskeren	Cloud9	3341
Blue	Mid	Nisqy	Cloud9	3719
Blue	ADC	Sneaky	Cloud9	3400
Blue	Support	Zeyzal	Cloud9	1990
Red	Top	V1per	FlyQuest	4205
Red	Jungle	Santorin	FlyQuest	3080
Red	Mid	Pobelter	FlyQuest	3193
Red	ADC	WildTurtle	FlyQuest	2924
Red	Support	JayJ	FlyQuest	2034
Blue	Team	Team	Cloud9	16118
Red	Team	Team	FlyQuest	15436

Data (Pre- and Post-Transform)

Side	Position	Player	Team	Gold at 10 Minutes
Blue	Top	Licorice	Cloud9	3668
Blue	Jungle	Svenskeren	Cloud9	3341
Blue	Mid	Nisqy	Cloud9	3719
Blue	ADC	Sneaky	Cloud9	3400
Blue	Support	Zeyzal	Cloud9	1990
Red	Top	V1per	FlyQuest	4205
Red	Jungle	Santorin	FlyQuest	3080
Red	Mid	Pobelter	FlyQuest	3193
Red	ADC	WildTurtle	FlyQuest	2924
Red	Support	JayJ	FlyQuest	2034
Blue	Team	Team	Cloud9	16118
Red	Team	Team	FlyQuest	15436

resultB	topB_gdat10	topB_gdat15	adcB_csdat10	adcB_csdat15
1	10	-481	12	0
1	244	647	-5	-11
1	185	350	19	16
0	720	1047	0	-3

Data



- Red represents a red win, while blue represents a blue win.

Classification Problem

- $\mathcal{D} = ((x_1, y_1), \ldots, (x_n, y_n))$ is the *design*.
  - $y_i \in \{0, 1\}$ is our binary response for a given $x_i \in \mathbb{R}^p$.
- Goal is to produce a binary prediction y based on a location x, based on the observed design $\mathcal{D}$, modeled as a probability

$$p(x) \equiv p(y = 1|\mathcal{D}).$$

Classification Problem

- $\mathcal{D} = ((x_1, y_1), \ldots, (x_n, y_n))$ is the *design*.
  - $y_i \in \{0, 1\}$ is our binary response for a given $x_i \in \mathbb{R}^p$.
- Goal is to produce a binary prediction y based on a location x, based on the observed design $\mathcal{D}$, modeled as a probability

$$p(x) \equiv p(y = 1|\mathcal{D}).$$

- The resulting model is what we call a classifier.
- Assign a class based off of a threshold, η.
  - $p(x) > \eta$ corresponds to the prediction $\hat{y} = 1$, else $\hat{y} = 0$.
  - We use $\eta = 0.5$, typical in many binary classification problems.
- We wish to assess out of sample performance on a general data set $\mathcal{D}' = ((x'_1, y'_1), \ldots, (x'_m, y'_m))$, where each $x'_i \notin \mathcal{D}$, so y'_i is unknown.

Classification Metrics - Accuracy

- *Accuracy* is the percentage of correct predictions,

$$\text{Acc} = \frac{\sum_{i=1}^{n} \mathbb{1}_{\{y_i = \hat{y}_i\}}}{n}$$

- We want this number to be as close to 1 as possible.
- In sports and esports literature, Accuracy is a regularly reported metric to assess a classifier.

Classification Metrics - Basic Terms

- A *False Positive* (FP) is when a positive result is predicted, but the true result is a negative.
  - $\text{FP}_i = \mathbb{1}_{\{y_i' \neq \hat{y}_i',\ y_i'=0\}}$
- A *False Negative* (FN) is when a negative result is predicted, but the true result is a positive.
  - $\text{FN}_i = \mathbb{1}_{\{y_i' \neq \hat{y}_i',\ y_i'=1\}}$
- A *True Positive* (TP) is when a positive result is predicted, and the true result is also a positive.
  - $\text{TP}_i = \mathbb{1}_{\{y_i' = \hat{y}_i',\ y_i'=1\}}$
- A *True Negative* (TN) is when a negative result is predicted, and the true result is also a negative.
  - $\text{TN}_i = \mathbb{1}_{\{y_i' = \hat{y}_i',\ y_i'=0\}}$

Classification Metrics - Basic Terms

- The *False Positive Rate* (FPR) is the ratio of false positives to the total number of negatives.

$$\text{FPR} = \frac{\sum_{i=1}^{m} \text{FP}_i}{\sum_{i=1}^{m} [\text{FP}_i + \text{TN}_i]} = \frac{\sum_{i=1}^{m} \mathbb{1}_{\{\hat{y}_i' = 1, y_i' = 0\}}}{\sum_{i=1}^{m} \left[\mathbb{1}_{\{\hat{y}_i' = 1, y_i' = 0\}} + \mathbb{1}_{\{\hat{y}_i' = y_i' = 0\}} \right]}$$

Classification Metrics - Basic Terms

- The *False Positive Rate* (FPR) is the ratio of false positives to the total number of negatives.

$$\text{FPR} = \frac{\sum_{i=1}^{m} \text{FP}_i}{\sum_{i=1}^{m} [\text{FP}_i + \text{TN}_i]} = \frac{\sum_{i=1}^{m} \mathbb{1}_{\{\hat{y}_i' = 1, y_i' = 0\}}}{\sum_{i=1}^{m} \left[\mathbb{1}_{\{\hat{y}_i' = 1, y_i' = 0\}} + \mathbb{1}_{\{\hat{y}_i' = y_i' = 0\}} \right]}$$

- The *True Positive Rate* (TPR), sometimes called *sensitivity*, is the ratio of true positives to the total number of positives.

$$\text{TPR} = \frac{\sum_{i=1}^{m} \text{TP}_i}{\sum_{i=1}^{m} [\text{TP}_i + \text{FN}_i]} = \frac{\sum_{i=1}^{m} \mathbb{1}_{\{\hat{y}_i' = y_i' = 1\}}}{\sum_{i=1}^{m} \left[\mathbb{1}_{\{\hat{y}_i' = y_i' = 1\}} + \mathbb{1}_{\{\hat{y}_i' = 0, y_i' = 1\}} \right]}$$

Classification Metrics - Basic Terms

- The *False Positive Rate* (FPR) is the ratio of false positives to the total number of negatives.

$$\text{FPR} = \frac{\sum_{i=1}^{m} \text{FP}_i}{\sum_{i=1}^{m} [\text{FP}_i + \text{TN}_i]} = \frac{\sum_{i=1}^{m} \mathbb{1}_{\{\hat{y}_i' = 1, y_i' = 0\}}}{\sum_{i=1}^{m} \left[\mathbb{1}_{\{\hat{y}_i' = 1, y_i' = 0\}} + \mathbb{1}_{\{\hat{y}_i' = y_i' = 0\}} \right]}$$

- The *True Positive Rate* (TPR), sometimes called *sensitivity*, is the ratio of true positives to the total number of positives.

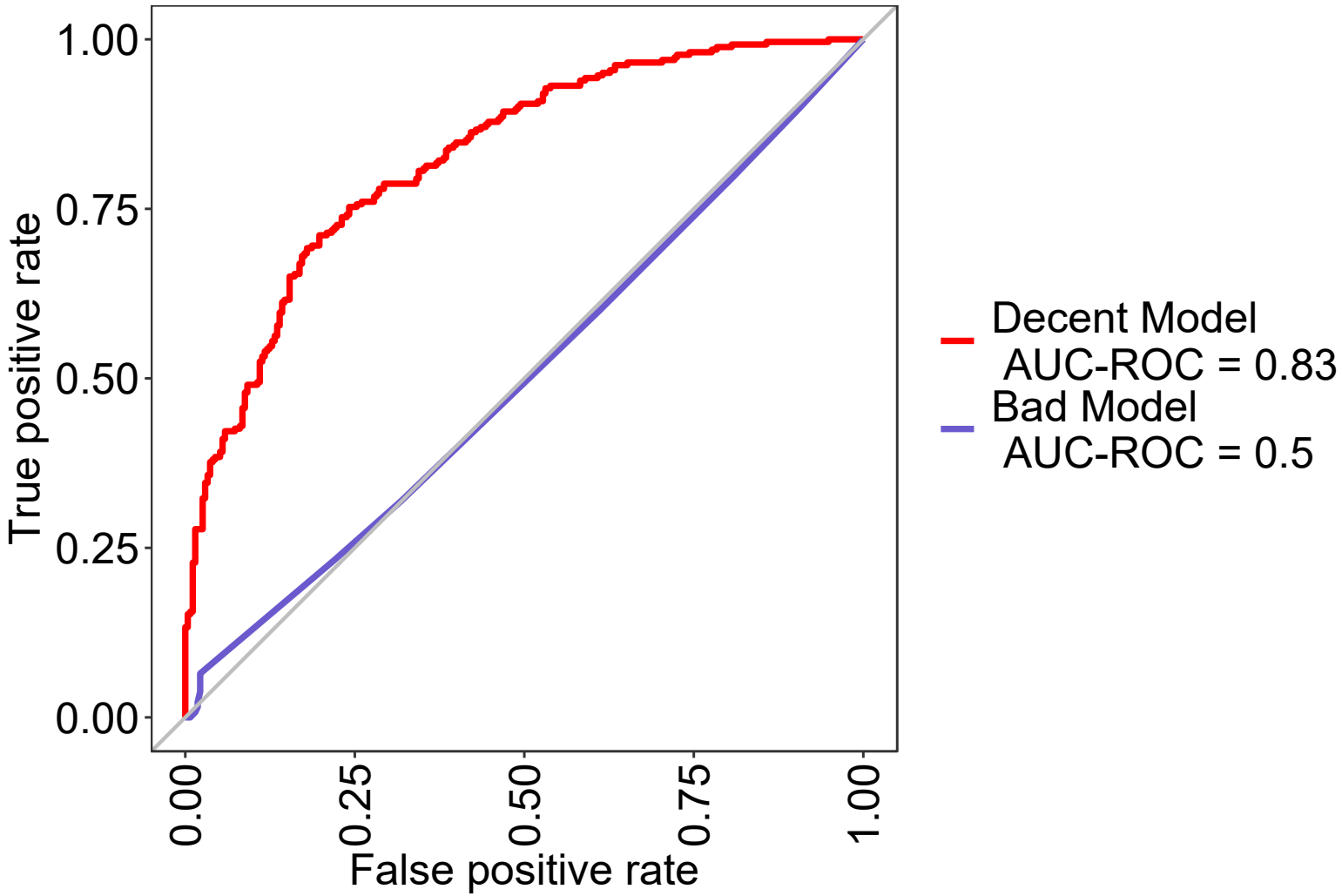
$$\text{TPR} = \frac{\sum_{i=1}^{m} \text{TP}_i}{\sum_{i=1}^{m} [\text{TP}_i + \text{FN}_i]} = \frac{\sum_{i=1}^{m} \mathbb{1}_{\{\hat{y}_i' = y_i' = 1\}}}{\sum_{i=1}^{m} \left[\mathbb{1}_{\{\hat{y}_i' = y_i' = 1\}} + \mathbb{1}_{\{\hat{y}_i' = 0, y_i' = 1\}} \right]}$$

- The *specificity* is defined as $1 - \text{FPR}$, which measures the proportion of actual negatives that are correctly identified as such.
- A classifier should aim to minimize the FPR (equivalent to maximizing the specificity) and maximize the TPR.

Classification Metrics - AUC and ROC Curve

- The *Receiver Operator Characteristic* (ROC) curve plots the FPR on the horizontal axis and the TPR on the vertical axis.
- The hope is that the TPR increases as rapidly as possible as the FPR increases, maximizing the *area under the ROC curve* (AUC).
- A good classifier's ROC curve will hug the top left of the plot.
- AUC shows how a classifier performs *overall* across various thresholds of η.

Classification Metrics - AUC and ROC Curve



Classification Metrics - Out of Sample Prediction

- We did an 80%/20% training/test set split.
- The set $\mathcal{D} = ((x_1, y_1), \ldots, (x_n, y_n))$ will be the training set.
- The set $\mathcal{D}' = ((x_1', y_1'), \ldots, (x_m', y_m'))$ will be the test set, with $\mathcal{D} \cap \mathcal{D}' = \emptyset$.
- The model will be fitted (trained) on $\mathcal{D}$, and evaluated on $\mathcal{D}'$.
- We can still assess the model on $\mathcal{D}$, but this can result in overfitting.

Classification Metrics - Cross-Validation

- Repeatedly emulates the training/test set split on $\mathcal{D}$ and averages the results.
- We used K-fold cross validation:
  - Split $\mathcal{D}$ into K non-overlapping parts.
  - All parts but one are used as a new training set, the one remaining part is the test set.
  - Repeat until all K folds are used as a test set, averaging the metric we evaluate on.

Classification Metrics - Cross-Validation

- Repeatedly emulates the training/test set split on $\mathcal{D}$ and averages the results.
- We used K-fold cross validation:
  - Split $\mathcal{D}$ into K non-overlapping parts.
  - All parts but one are used as a new training set, the one remaining part is the test set.
  - Repeat until all K folds are used as a test set, averaging the metric we evaluate on.
- Larger K means longer but more accurate computations.
- When $K = n$, this is called leave-one-out cross-validation (LOOCV).
- Typically, K is chosen as 5 or 10 as they give similar results to that of LOOCV with significantly shorter computation time, we used $K = 10$.

Classification Models

- Reminder: We wish to create a classifier that typically minimizes some loss criteria utilizing $\mathcal{D}$, modeled as a probability with

$$p(x) \equiv p(y = 1|\mathcal{D}).$$

- The previous metrics offer ways to assess and quantify loss, but do not produce a classifier.
- The next step is to determine possible classifiers to test.

Classification Models - Logistic Regression

- Defines $p(x)$ implicitly in terms of a generalized log-odds or *logit* function,

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p, = \beta_0 + \beta^T x,$$

  with $x = (x_1, \ldots, x_p)$ and $\beta = (\beta_1, \ldots, \beta_p)$.
- The above equation can be re-written as

$$p(x) \equiv p(x;\beta) = \frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}.$$

- Each y_i is Bernoulli distributed with probability of success $p(x_i)$:

$$p_{y_i|\mathcal{D}}(y_i) = p(x_i)^{y_i}(1-p(x_i))^{1-y_i}.$$

- Estimates coefficients with maximum likelihood by maximizing the log-likelihood function

$$\ell(\beta) = \sum_{i=1}^{n}\left\{y_i(\beta_0+\beta^T x_i) - \log(1+e^{\beta_0+\beta^T x_i})\right\}.$$

  - Has no closed form maximum, so optimal β must be solved for numerically.

Classification Models - LASSO

- Same formulation as logistic regression, but induces the ℓ_1 penalty on β, shrinking some coefficients to 0:

$$\max_{\beta_0,\beta} \left\{ \sum_{i=1}^{n} \left[y_i(\beta_0 + \beta^T x_i) - \log(1 + e^{\beta_0+\beta^T x_i}) \right] - \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

  - λ is the penalty term for the ℓ_1 regularization.
  - λ is tuned by cross-validation on different metrics.
  - Larger β values will reduce the quantity inside, lowering the "maximum".
  - There is a tradeoff between increasing β and trying to maximize the likelihood.
- Results in a form of "automatic" model and feature selection.

Classification Models - LDA

- Aims to draw the best linear decision boundary by:
  - Proposing a conditional distribution for the predictors, $x_i|y_i, i = 1, \ldots, n$.
  - Placing a prior on each y_i.
  - Using Bayes' theorem to obtain $p(x)$.
- $\pi_k, k = 0, 1$ is the prior distribution for each y_i.
- $q_k(x) = p(x|y = k)$ is the conditional distribution of x whose corresponding y value is k.
- Then, Bayes' theorem states

$$p(x) = p(y = 1|x) = \frac{\pi_1 q_1(x)}{\pi_0 q_0(x) + \pi_1 q_1(x)}.$$

Classification Models - LDA

- Assumes that $x = (x_1, \ldots, x_p)$ is drawn from a MVN distribution with:
  - Mean vector $\mu = (\mu_1, \ldots, \mu_p)$ where $\mu_i = \mathbb{E}[x_i]$.
  - A covariance matrix that is common to all classes, Σ, with i, j entry as $\text{cov}(x_i, x_j)$.

Classification Models - LDA

- Assumes that $x = (x_1, \ldots, x_p)$ is drawn from a MVN distribution with:
  - Mean vector $\mu = (\mu_1, \ldots, \mu_p)$ where $\mu_i = \mathbb{E}[x_i]$.
  - A covariance matrix that is common to all classes, Σ, with i, j entry as $\text{cov}(x_i, x_j)$.
- MVN density is defined as

$$q(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

- Plugging the density function for the kth class into Bayes' theorem reveals that the LDA classifier assigns an observation x to the class k for which

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log(\pi_k) \qquad k = 0, 1$$

  is largest, where μ_k is a class-specific mean vector.
  - This equation is linear in x, which is what results in the linear decision boundary.

Classification Models - LDA

- In practice, we must estimate the parameters of the MVN distribution using $\mathcal{D}$:
  - $\hat{\pi}_k = n_k/n$, where n_k is the number of class-k observations
  - $\hat{\mu}_k = \sum_{y_i=k} x_i/n_k$, where $\sum_{y_i=k}$ represents the sum over all indices such that $y_i = k$
  - $\hat{\Sigma} = \left[\sum_{y_i=0}(x_i - \hat{\mu}_0)(x_i - \hat{\mu}_0)^T + \sum_{y_i=1}(x_i - \hat{\mu}_1)(x_i - \hat{\mu}_1)^T \right] /(n-2)$

Classification Models - QDA

- Very similar to that of LDA, but now assumes each class has its own covariance matrix.
- Again, plugging the MVN density into Bayes' theorem, the QDA classifier assigns an observation x to the class k for which

$$\delta_k(x) = -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\log|\Sigma_k| + \log(\pi_k)$$

  is largest, where Σ_k is the covariance matrix for class k.
  - Now, the equation is quadratic in x, resulting in a quadratic decision boundary.

Classification Models - Maximal Margin Hyperplane

- Goal is to create a separating hyperplane from the training observations.
- A p-dimensional hyperplane for $x \in \mathbb{R}^p$ has the form

$$\{x : f(x) = x^T\beta + \beta_0 = 0\},$$

  where $||\beta|| = 1$.
- For x not in the hyperplane, we have either $f(x) > 0$ or $f(x) < 0$, indicating which side of the hyperplane that x is on.
  - Leads to a classification rule induced by $f(x)$:

$$G(x) = \text{sign}[x^T\beta + \beta_0].$$

  - Directly provides the value of $\hat{y}$ as -1 if $G(x) < 0$, and 1 if $G(x) > 0$.
  - Coding is equivalent to our $y \in \{0, 1\}$ used previously.

Classification Models - Maximal Margin Hyperplane

- Generally impossible to construct a classifier $\hat{G}(x)$ based on a hyperplane $\hat{f}(x)$ so that each observation produces $\hat{G}(x) = y_i$.
- We introduce:
  - A cost parameter C which represents how much we are willing deviate from the hyperplane.
  - Slack variables ξ_i, where $0 \leq \xi_i \leq 1$ for each $i = 1, \ldots, n$.

Classification Models - Maximal Margin Hyperplane

- Generally impossible to construct a classifier $\hat{G}(x)$ based on a hyperplane $\hat{f}(x)$ so that each observation produces $\hat{G}(x) = y_i$.
- We introduce:
  - A cost parameter C which represents how much we are willing deviate from the hyperplane.
  - Slack variables ξ_i, where $0 \leq \xi_i \leq 1$ for each $i = 1, \ldots, n$.
- C is specified ahead of time, whle ξ_i are part of the minimization procedure:

$$\min_{\beta, \beta_0} \frac{1}{2} \sum_{j=1}^{p} \beta_j^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{subject to } \xi_i \geq 0, y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i \ \forall i.$$

- Solution to the minimization problem can be found by finding the Lagrange function.

Classification Models - Maximal Margin Hyperplane

- Alternatively, a specific solution can be found by applying the Kuhn-Tucker theorem:

$$\hat{\beta} = \sum_{i=1}^{n} \hat{\alpha}_i y_i x_i$$

- Remarkably, this results in a fitted hyperplane:

$$\hat{f}(x) = x^T \hat{\beta} + \hat{\beta}_0 = \sum_{i=1}^{n} \hat{\alpha}_i y_i \langle x, x_i \rangle + \hat{\beta}_0,$$

a closed form involving the inner products of x and the training points x_i.

Classification Models - SVMs

- SVMs are extensions of the above which use kernel functions that associate "closeness" between x and x_i rather than the dot product above.
- Kernels are motivated by certain shapes in the observed data.
- Now, with the kernel replacing the dot product, we get the classifier $\hat{G}(x) = \text{sign}[\hat{f}(x)]$ where

$$\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0.$$

Classification Models - SVMs

- There are three types of kernels that we were interested in:
  - The linear kernel:

$$K(x, x') = \langle x, x' \rangle.$$

Classification Models - SVMs

- There are three types of kernels that we were interested in:
  - The linear kernel:
    $$K(x, x') = \langle x, x' \rangle.$$
  - The polynomial kernel of degree d:
    $$K_p(x, x') = (\langle x, x' \rangle + c)^d$$
    with d being a positive integer greater than 1.

Classification Models - SVMs

- There are three types of kernels that we were interested in:
  - The linear kernel:
$$K(x, x') = \langle x, x' \rangle.$$
  - The polynomial kernel of degree d:
$$K_p(x, x') = (\langle x, x' \rangle + c)^d$$
with d being a positive integer greater than 1.
  - The radial kernel:
$$K_r(x, x') = \exp\left(-\gamma \sum_{j=1}^{p} (x_j - x_j')^2\right)$$
where γ is a positive constant determining the "spread" of the shape.

Classification Models - SVMs

- There are three types of kernels that we were interested in:
  - The linear kernel:
$$K(x, x') = \langle x, x' \rangle.$$
  - The polynomial kernel of degree d:
$$K_p(x, x') = (\langle x, x' \rangle + c)^d$$
with d being a positive integer greater than 1.
  - The radial kernel:
$$K_r(x, x') = \exp\left(-\gamma \sum_{j=1}^{p} (x_j - x_j')^2\right)$$
where γ is a positive constant determining the "spread" of the shape.
- The polynomial kernel is a much more flexible decision boundary than the linear kernel as it has a decision boundary of the form of whatever degree the polynomial is.
- The radial kernel has very local behavior, where only nearby training observations have an effect on the classification.

Classification Models - GPs

- A Gaussian process is a collection of random variables $(f(x))$, $x \in \mathbb{R}^p$ such that every finite collection of them has a MVN distribution.
  - Mean function is defined as $\mu(x) = \mathbb{E}[f(x)]$
  - Covariance function is defined as $C(x, x') = \text{cov}(f(x), f(x'))$.
- Treat f as an unknown function with input x, obtain posterior prediction given the data using its MVN properties.

Classification Models - GPs

- A Gaussian process is a collection of random variables $(f(x))$, $x \in \mathbb{R}^p$ such that every finite collection of them has a MVN distribution.
  - Mean function is defined as $\mu(x) = \mathbb{E}[f(x)]$
  - Covariance function is defined as $C(x, x') = \text{cov}(f(x), f(x'))$.
- Treat f as an unknown function with input x, obtain posterior prediction given the data using its MVN properties.
- The process for GP classification is as follows:
  1. Specify a prior mean and covariance function;
     - We use $\mu(x) = 0$ as we are not extrapolating and this greatly simplifies the fitting process.
     - C can be any positive definite symmetric function $C : \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$.
  2. Let y_x be the binary response for a general x that is given, and $f_x \equiv f(x)$ be the GP evaluated at x, meaning $f(x)$ is univariate normal.
  3. The probabilities of y_x are then defined conditionally:

$$p(y_x = 1 | f_x) = \frac{1}{1 + e^{-f_x}}$$

Classification Models - GPs

- Goal is to produce a probability for prediction

$$p(x) = p(y = 1|\mathcal{D}) = p(y = 1|y_1, \ldots, y_n).$$

  - This is a binomial conditional on a binomial, but these two binomials are only defined conditionally on GPs.
  - Extremely difficult to compute as no closed form exists, so must resort to numerical techniques.
- MCMC can be used, but has a runtime complexity of n^3, taking a very long time to run.
- We look to the Laplace Approximation, which has a worst case runtime of $n^3/6$.
- Laplace Approximation saves much time, but may give a poor approximation to the true shape of the posterior.
- Usually, the risk is not a big concern as the results are still similar and time saved is drastic.

Results - Fitting and Diagnostics

- Metrics used for fitting will be AUC and 10-Fold CV Accuracy.
  - We give a bit more value to 10-Fold CV Accuracy than AUC.
- The above metrics will be given for all of the modeling techniques we used: logistic regression, LDA, QDA, LASSO, SVMs, and GPs.

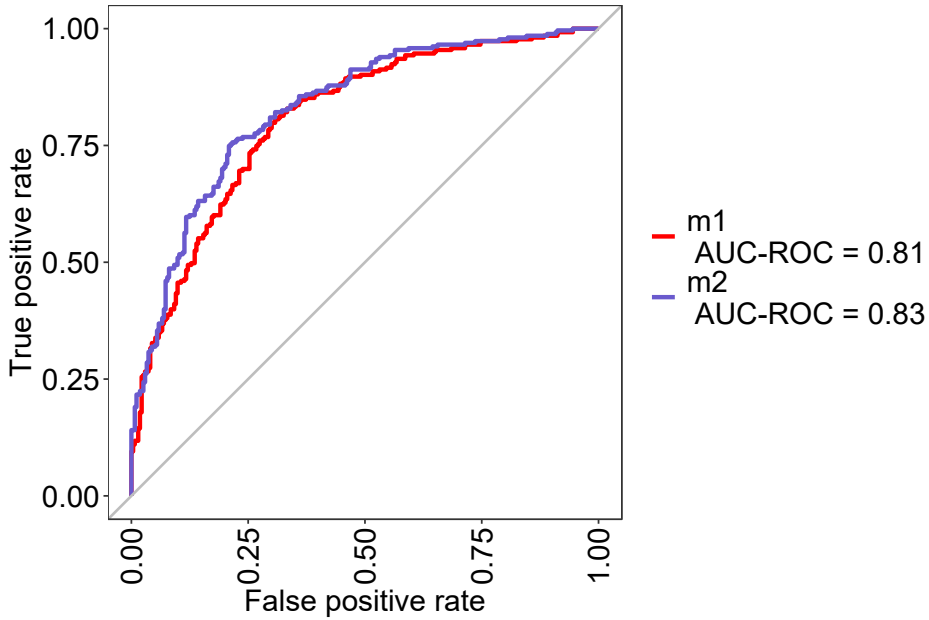
Results - Fitting and Diagnostics

- Metrics used for fitting will be AUC and 10-Fold CV Accuracy.
  - We give a bit more value to 10-Fold CV Accuracy than AUC.
- The above metrics will be given for all of the modeling techniques we used: logistic regression, LDA, QDA, LASSO, SVMs, and GPs.
- We use subsets of predictors from forward stepwise selection (FSS) on logistic regression based on the two metrics above for every model aside from LASSO and SVMs.
- Models resistant to overfitting such as SVMs and GPs also use all predictors.

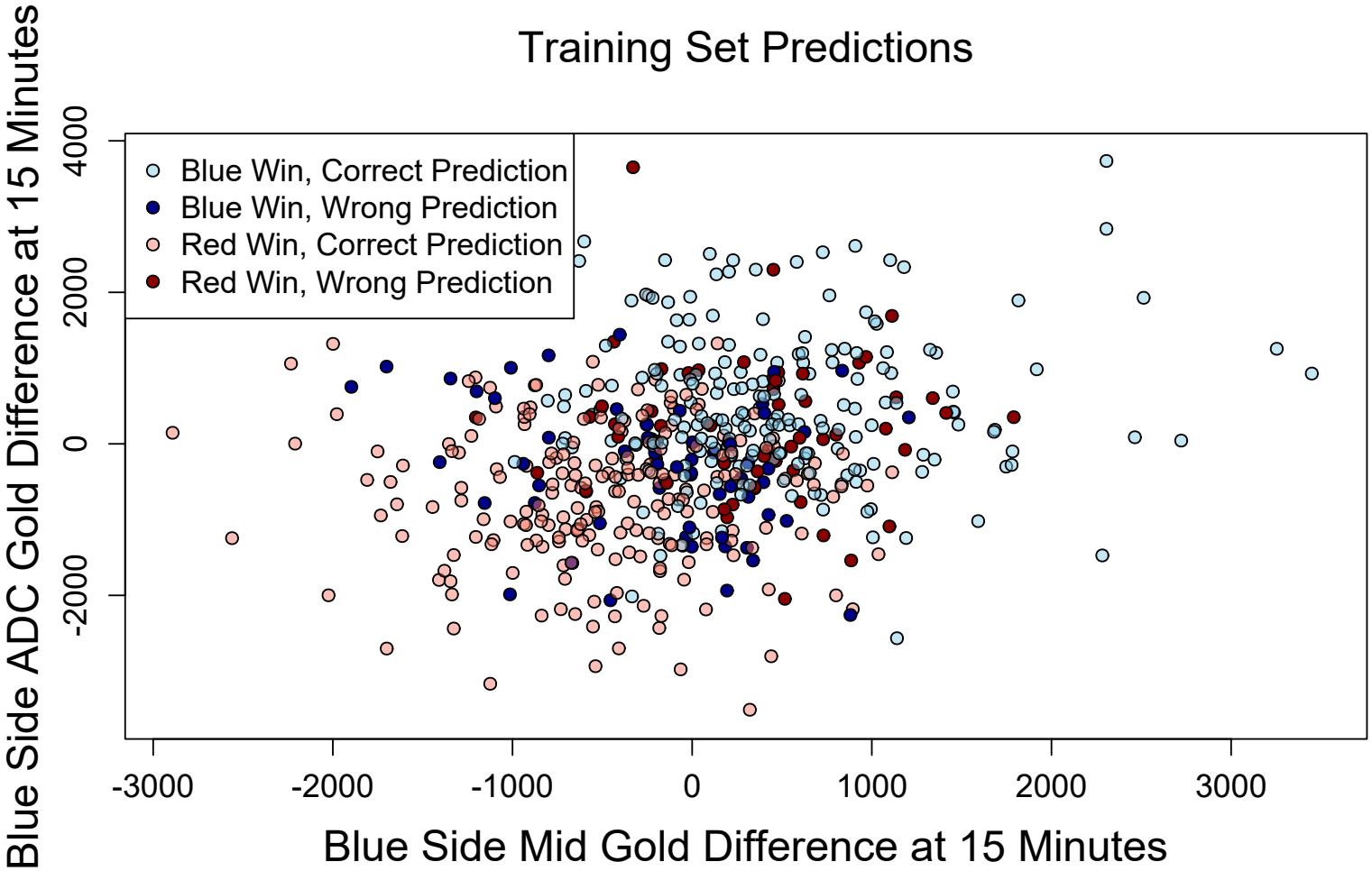
Results - Logistic Regression

- Models were obtained from FSS on the basis of Acc (m1) and AUC (m2).

Model	10-Fold CV Acc	AUC
(m1)	0.7799	0.81
(m2)	0.7647	0.83



Results - Logistic Regression

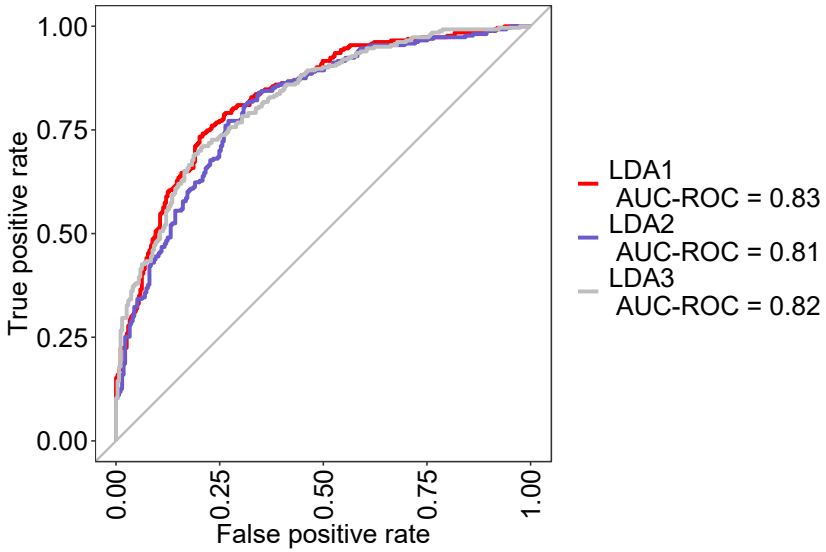


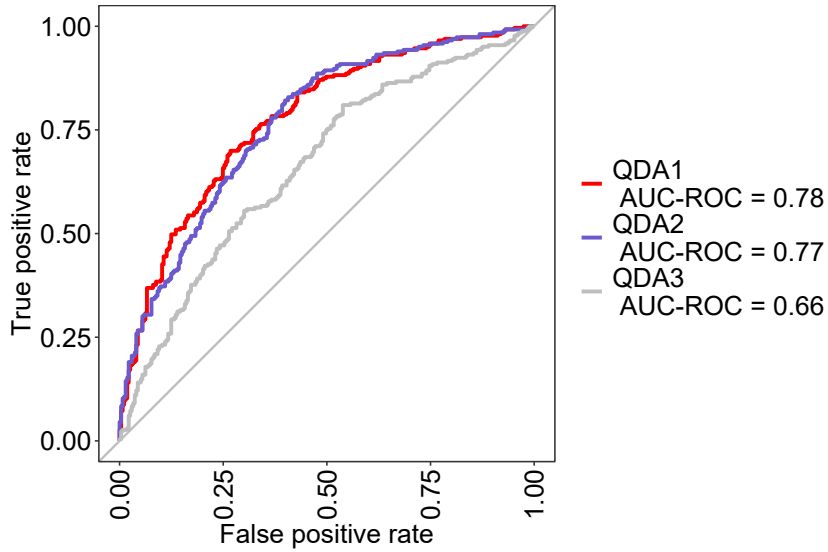
Results - LDA/QDA

- Three of each LDA and QDA models were fit:
  - (LDA1) and (QDA1) used the predictors from (m1) in logistic regression (forward subset selection on Acc).
  - (LDA2) and (QDA2) used the predictors from (m2) in logistic regression (forward subset selection on AUC).
  - (LDA3) and (QDA3) used all predictors.

Results - LDA/QDA

Model	10-Fold CV Acc	AUC
(LDA1)	0.7609	0.83
(LDA2)	0.7347	0.81
(LDA3)	0.7423	0.82
(QDA1)	0.7144	0.78
(QDA2)	0.6937	0.77
(QDA3)	0.6192	0.66





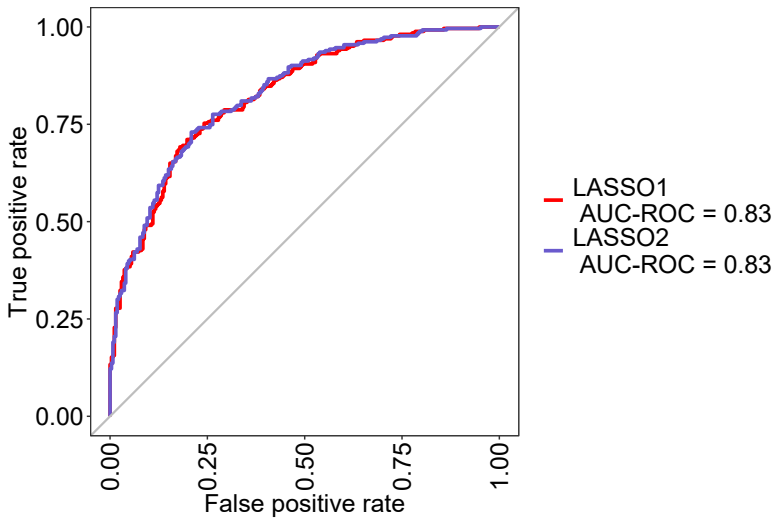
Results - LASSO

- Began with all predictors, and obtained the optimal λ values on the two different metrics using cross-validation:
  - (LASSO1) had its λ chosen on the basis of Acc, resulted in all predictors except one, so $p = 27$.
  - (LASSO2) had its λ chosen on the basis of AUC, resulted in 21 predictors, so $p = 21$.

Results - LASSO

- Began with all predictors, and obtained the optimal λ values on the two different metrics using cross-validation:
  - (LASSO1) had its λ chosen on the basis of Acc, resulted in all predictors except one, so $p = 27$.
  - (LASSO2) had its λ chosen on the basis of AUC, resulted in 21 predictors, so $p = 21$.

Model	10-Fold CV Accuracy	AUC	λ
(LASSO1)	0.7535	0.83	0.0040
(LASSO2)	0.7501	0.83	0.0040

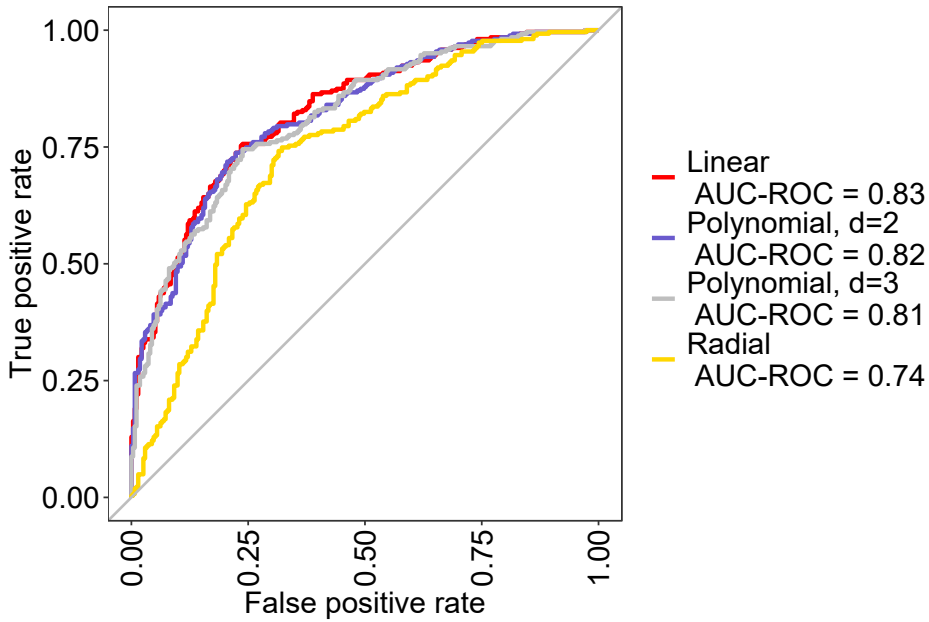


Results - SVMs

- All SVMs had their cost tuned from $c = \{0.1, 0.25, 0.5, 1, 1.5, 2, 4, 8, 10, 16, 25, 32, 64, 100\}$.
- The radial kernel also had the extra hyperparameter γ tuned from $\gamma = \{0.5, 1, 2, 3, 4\}$.

Results - SVMs

Kernel	10-Fold CV Acc	AUC
Linear	0.7591	0.83
Polynomial with $d = 2$	0.7552	0.82
Polynomial with $d = 3$	0.7515	0.81
Radial	0.7050	0.74

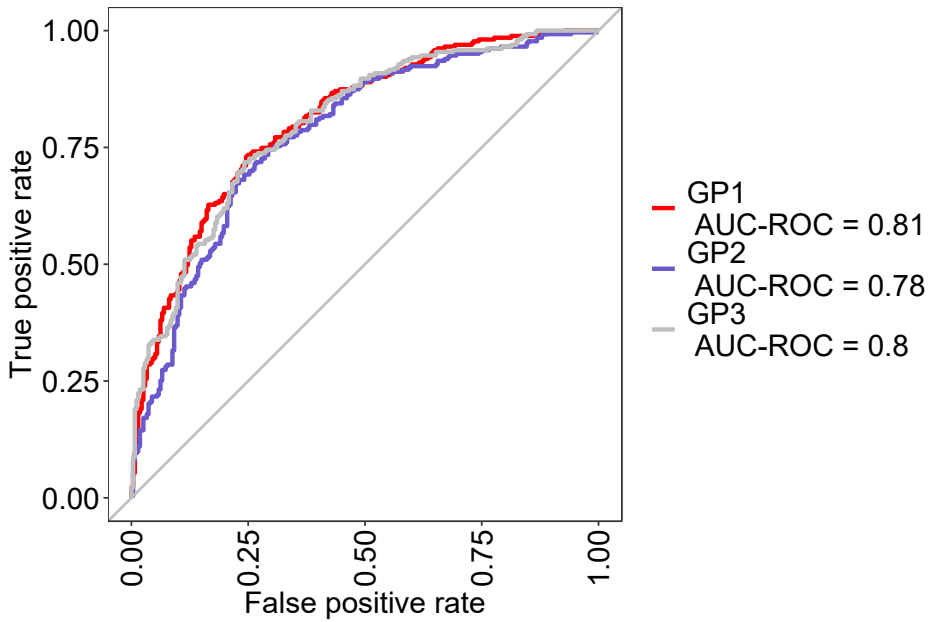


Results - GPs

- Three GPs were fit, all using the Gaussian (radial) kernel.
  - (GP1) uses the same predictors as that of (m1) from logistic regression (FSS on Acc).
  - (GP2) uses the same predictors as that of (m2) from logistic regression (FSS on AUC).
  - (GP3) uses all predictors.

Results - GPs

Model	10-Fold CV Acc	AUC
(GP1)	0.7330	0.81
(GP2)	0.7216	0.78
(GP3)	0.7320	0.80



Results - Training Set

Method	Model	10-Fold CV Acc	AUC
Logistic Regression	(m1)	0.7647	0.81
LDA	(LDA1)	0.7609	0.83
QDA	(QDA1)	0.7144	0.78
LASSO	(LASSO1)	0.7535	0.83
SVM	(SVM-Lin)	0.7591	0.83
Gaussian Process	(GP1)	0.7330	0.81

Results - Training Set

Method	Model	10-Fold CV Acc	AUC
Logistic Regression	(m1)	0.7647	0.81
LDA	(LDA1)	0.7609	0.83
QDA	(QDA1)	0.7144	0.78
LASSO	(LASSO1)	0.7535	0.83
SVM	(SVM-Lin)	0.7591	0.83
Gaussian Process	(GP1)	0.7330	0.81

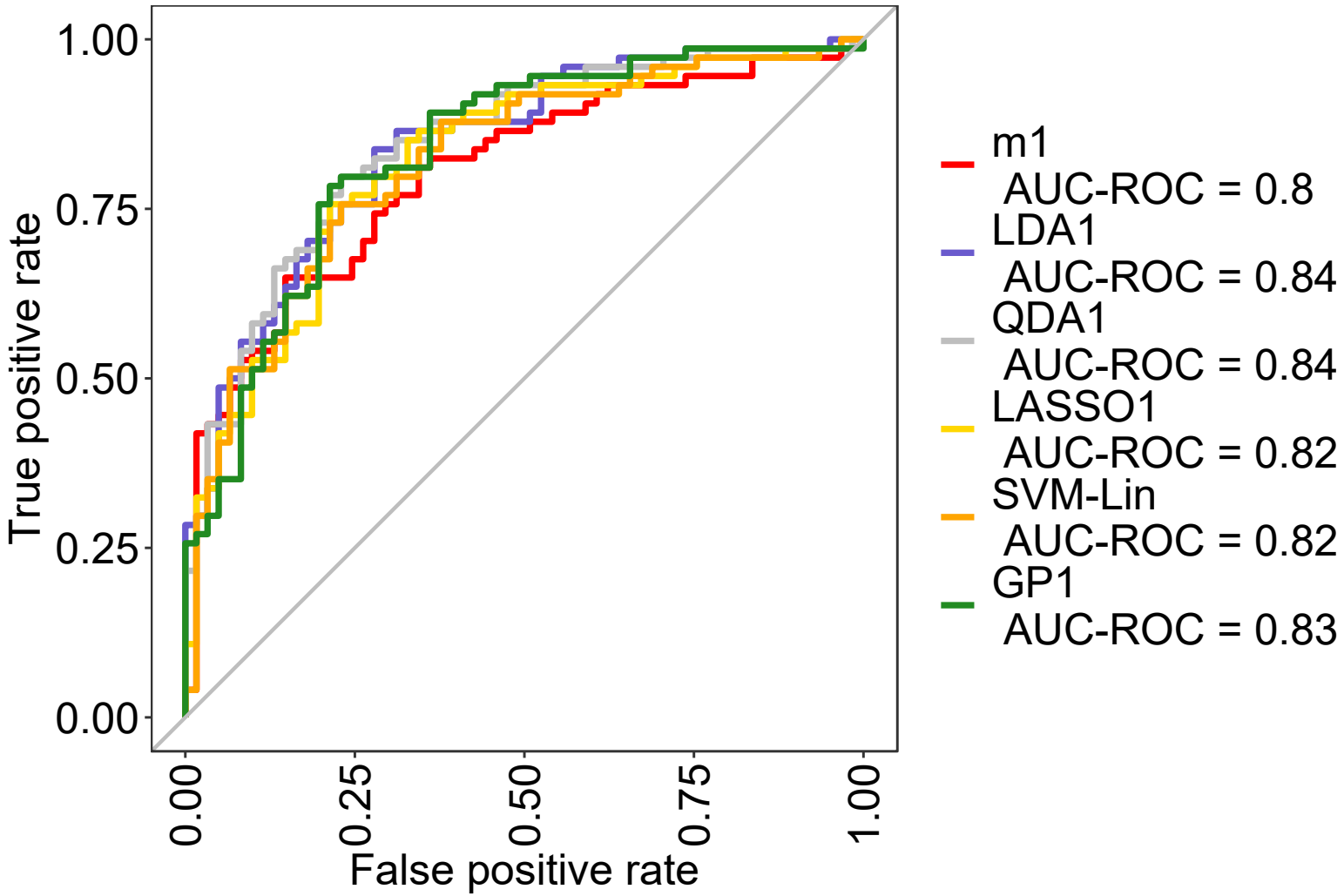
Next, each of these models will be evaluated on the test set.

Results - Test Set

Method	Model	Acc	AUC
Logistic Regression	(m1)	0.7556	0.80
LDA	(LDA1)	0.7556	0.84
QDA	(QDA1)	0.7704	0.84
LASSO	(LASSO1)	0.7481	0.82
SVM	(SVM-Lin)	0.7333	0.82
Gaussian Process	(GP1)	0.7852	0.83

- (LDA1) tied for the best AUC on both the training and test set, with good accuracy as well.
- (m1) and (LASSO1) stay fairly consistent across the two sets.
- (GP1) performed somewhat poorly on the training set, but had the best accuracy on the test set and was a strong contender for AUC as well.
- (QDA1), which performed the worst on the training set for both metrics, had the best AUC and second best accuracy on the test set.

Results - Test Set



Conclusion

- Hard to determine a "best" model.
- Theory of GPs seems fit for the domain of the problem.

Conclusion

- Hard to determine a "best" model.
- Theory of GPs seems fit for the domain of the problem.
- We thought QDA would catch the finer points that LDA could not, but that did not seem to be the case at least on the training set.

Conclusion

- Hard to determine a "best" model.
- Theory of GPs seems fit for the domain of the problem.
- We thought QDA would catch the finer points that LDA could not, but that did not seem to be the case at least on the training set.
- Some of the inconsistencies in training and test set results may have come from:
  - Sizes of the sets: training set had $n = 536$ observations, while the test set had $m = 135$ observations.
  - 10 folds may not have been enough to estimate test set accuracy.

Conclusion - Future Work

- Access to Riot Games' API to pull data minute-by-minute, allowing for live predictions.
  - This would also allow access to high-level matches, not just professional matches.

Conclusion - Future Work

- Access to Riot Games' API to pull data minute-by-minute, allowing for live predictions.
  - This would also allow access to high-level matches, not just professional matches.
- Incorporate champions picked per role, player names, and interactions between these factors.
  - These are fixed pre-match, but need much more data as some players and champions only appear a handful of times.

Conclusion - Future Work

- Access to Riot Games' API to pull data minute-by-minute, allowing for live predictions.
  - This would also allow access to high-level matches, not just professional matches.
- Incorporate champions picked per role, player names, and interactions between these factors.
  - These are fixed pre-match, but need much more data as some players and champions only appear a handful of times.
- Handpick sets of predictors for models outside of logistic regression.
- Consider other popularly used methods such as random forests, neural networks, gradient-boosted trees, etc.

Conclusion - Final Words

- Broadcasts currently lack match prediction statistics.

Conclusion - Final Words

- Broadcasts currently lack match prediction statistics.
- Not much literature for LoL yet, regardless of being such a large esport.

Conclusion - Final Words

- Broadcasts currently lack match prediction statistics.
- Not much literature for LoL yet, regardless of being such a large esport.
- Even with a fraction of possible predictors, we can accurately predict the result 78% of the time.

Conclusion - Final Words

- Broadcasts currently lack match prediction statistics.
- Not much literature for LoL yet, regardless of being such a large esport.
- Even with a fraction of possible predictors, we can accurately predict the result 78% of the time.
- Hopefully this can help serve as a guide to organizations and even companies to help improve their team's performance.

Thank You!

Thank you all so much for coming to this and allowing me to present.