CIS 520 Spring 2019, Project Report

Video Game Balance in ESports using ML: Professional vs. Casual *League of Legends* Matches

Team Members:

Sadat Shaik (PennKey: `sshaik`; Email: `sshaik@seas.upenn.edu`)
Ziad Ben Hadj-Alouane (PennKey: `ziadb`; Email: `ziadb@seas.upenn.edu`)
Ayya Elzarka (PennKey: `ayyaelz`; Email: `ayyaelz@wharton.upenn.edu`)
William Corse (PennKey: `wcorse`; Email: `wcorse@seas.upenn.edu`)

Assigned Project Mentor:

Barry Plunkett

Team Member Contributions:

Team Member	Contributions
Sadat Shaik	Neural Network, Data Cleaning and Processing, Writing Proposal Model Evaluation / Visualization, Writing Project Report
Ziad Ben Hadj-Alouane	Literature Review, Writing Proposal, Writing Project Report Preparing Presentation Linear SVM, Kernel SVM
Ayya Elzarka	Random Forest Data Processing, Writing Project Report
William Corse	Logistic Regression, Literature Review, Writing Project Proposal Data Validation, Writing Project Report, Graphs

Code Submission:

Code is submitted via Canvas as a .zip file of .ipynb notebooks.

Abstract

League of Legends (LoL) is an online competitive video game where two teams of five players each battle against each other. In this report we attempt to answer two questions: before a match begins, can we predict the winning team? Does our classification accuracy differ between two types of datasets: professional player data, and casual player data? We explored these questions by modeling pre-match predictions using binary classification methods, each evaluated on both datasets. We found that our models were able to predict match outcomes in casual settings far more accurately than in professional settings.

1 Introduction

Video game companies expend massive resources fine-tuning their games to make them fair and balanced[1] for their players. However, they might do so in different ways depending on their player base. Specifically, the game is usually played differently in the professional [2] scene than in the casual [3] scene. This report investigates balance in *League of Legends*, a popular online competitive video game, by attempting to answer two questions: before a match begins and given pre-game data, can we predict the winning team? Does our classification accuracy differ between professional and casual Matches? If so, why?

2 Related Work

Online forums, as seen in [4], measure match prediction accuracy without in-game data using a trained neural network, while [3] predicts match outcomes in five minute intervals using in-game data and random forest classifiers. The former exclusively relies on team character selections, while the latter focuses on in-game features. [6] combines player match-up data and in-game statistics, and measures prediction accuracy as a function of game time. Hodge et al., analyzed a different video game (DotA), and conclude that casual data can be used to predict professional data outcomes, at the cost of slightly reduced accuracy (3%) [5]. Yang et al. measure real-time match prediction results, and demonstrate that the more pre-game data they include in their model, the more accurate its predictions [7].

3 Data Set

The casual data set includes 144k casual matches [4] collected from Kaggle. Of note is the percentage of matches in which the blue team wins: 49.9%. Our professional data set includes 7.3k ESports

[1] **Game balance** refers to the fairness in the gameplay itself: does the game reward skill and general player ability? If so, the game is considered balanced.

[2] **Professional** scene means the subset of matches played that only involve professional and highly skilled players that play the game for a living. The players in considerations are part of the ESports community

[3] **Casual** scene refers to the subset of matches from players all over the world that only play the game for fun.

[4] In the casual dataset, players start a "solo" game where they are automatically matched with 9 other randomly selected players of "similar" skill.

matches also collected from Kaggle. Interestingly, the blue team wins in 54.715% of professional matches [1]. Tab. [1] and Fig. [2] reveals an interesting trend in our casual data set – on average, the losing team in a casual match is at a lower average skill level. Additionally, on average, players are matched to a stronger opponent in the role they select to play. Finally, we note that on average, the difference between the most skilled player and the lowest skilled player on the losing team is higher than that on the winning team.

4 Problem Formulation

We consider a binary classification problem with instance space $\mathcal{X}$ and label space $\mathcal{Y} = \{\pm 1\}$ and underlying distribution D on $(\mathcal{X} \times \mathcal{Y})$. In our setting, the instance space constitutes LoL matches by only considering pre-game features such as champion picks, player skill, and champion winrates [5]. The goal is to learn a binary classifier that accurately predicts the winning team ($y = +1$ indicates the blue team winning).

We ran our algorithms on two datasets, with **each dataset coming from a different distribution:** professional data is generated from D_p on $(\mathcal{X}_p, \mathcal{Y})$, while casual is generated from D_c on $(\mathcal{X}_c, \mathcal{Y})$. $\mathcal{X}_p$ and $\mathcal{X}_c$ are subsets from the original instance space $\mathcal{X}$ that comprises the corpus of all LoL matches.

To ensure that we have a non-trivial classifier (i.e. one that doesn't heavily bias towards predicting one class), we chose the F_1 performance measure. By optimizing for both precision and recall, we ensure that the model is learning non-trivial characteristics to predict match outcomes.

Chronological Ordering: We chronologically order our datasets. Because player behavior can change over time (e.g. players can become more skilled), we ensure that we do not use players' future games to predict their past performance.

Categorical Feature 1-Hot Encoding: Our data set contains many categorical features (player names, selected champions, etc.), which are initially encoded as integers[6]. To avoid factoring in the magnitude of the encoding into the learned weights, we one-hot encode all categorical variables. The downside to this procedure is the large feature space that results (3634 and 2901 features for casual and professional datasets respectively).

Reducing # of Categorical Features: To reduce the dimensionality of our feature space after naively one-hot encoding all categorical variables, we collected statistics for some features. Specifically, instead of one-hot encoding lane matchups, we represented them with the average winrates of that lane matchup as a percentage [7]. Similarly, we replaced the one-hot encoding the player names in the professional dataset, with that player's individual winrate. To avoid encoding test data, we only compiled these statistics from the training set. Additionally, for any lane matchups not represented in our training data, we set the probability to 50% as we have no prior baseline.

Principal Component Analysis: As previously mentioned, 1-hot encoding categorical data

[5] **Winrate** refers to the percentage of matches a player or champion wins in the data set.

[6] Note that only the professional dataset provided information on each player (e.g. their ID)

[7] **Lane Matchups** refer to the fact that each player on each team fulfills a specific role (e.g. attacker, defender, etc.). Typically a player on the blue team will only interact with the player of the same role on the red team.

resulted in a large feature space. This slowed down most of our models by a significant factor. To remedy this issue, we used PCA to extract at least 90% of the variance, reducing the number of features to 154 for professional data (with winrates) and to 44 for casual data (with winrates). Fig. [10] shows the amount of seconds saved after running select algorithms with PCA-modified data.

5 Algorithms & Cross-Validation Parameters

Our codebase is written in Python. Our data processing pipeline was done using *pandas* and *numpy*. Implementing Logistic Regression, SVM, Kernel SVM, and Random Forests was done with *scikit-learn*. Neural Networks, were implemented using *PyTorch* to take advantage of GPU optimization.

Logistic Regression: Linear logistic regression is a low-complexity model and excels at detecting linearly-separable data in a binary classification setting. Our model includes a 5-fold cross-validated L_2 regularizer to avoid over-fitting. We searched over $L_2^{-1} = \{10^{-4}, 10^3, \ldots, 10^2\}$ and selected 10^{-3} for both datasets (see Fig. [3]). In place of 0-1 loss, we used a surrogate logistic loss function. We select a limited-memory BFGS solver to minimize empirical logistic training error, as it has been found to converge faster for some higher-dimensional training sets [2].

Linear and RBF SVMs: Like logistic regression, Linear SVM has low-complexity and is suitable for linearly-separable data. We performed 5-fold cross-validation to tune the hyperparameters. We specifically tune this parameter by using the F_1 as a measure of performance. For Linear SVM, the best parameter was $C = 1$ from the range $\{1, 5, 10, 100, 1000\}$ for both datasets. To capture non-linear properties of the data, we used Kernel SVM. Specifically, we used the RBF Kernel as it is a widely used option in practice due to its flexibility and low number of hyperparameters. This was especially useful for professional data where Linear SVM performed poorly. 5-fold cross-validation on $C \in \{1, 5, 10, 100, 1000\}$ and $\gamma \in \{10^{-3}, 10^{-4}, 10^{-5}\}$ parameters yielded $(\gamma = 0.0001, C = 5)$ for casual, and $(\gamma = 0.001, C = 1)$ for professional. In both SVM methods, we used the standard squared hinge loss function to assign penalties for misclassificaton.

Random Forests: We used random forests to control for overfitting as it is an ensemble method in which each tree is constructed on different training samples. The random forest was tuned using the out of bag score to cross validate, this is calculated by applying the model to the data that was held out for each decision tree. Parameters tuned were: # of trees $\{500, 1000, ..., 3000\}$, the max depth of the tree $\{3, 4, ..., 7\}$, the min number of samples to split an internal node $\{2, 3, ..., 7\}$, the min # of samples in a leaf node $\{1, 2, ..., 5\}$, etc. To encourage splitting on criterion that results in pure leaf nodes, impurity measures are minimized, rather than 0-1 errors when growing a decision tree. We selected the Gini Index as it is optimal when prioritizing reduction of mis-classification.

Neural Networks: Thus, Neural Networks were a good candidate as they are robust to large feature spaces which may be highly correlated. The network uses mini-batch gradient descent with batch normalization as a compromise between the slowness of batch gradient descent and the noisiness of SGD. We use ReLU as the non-linearity to prevent vanishing gradients except for the final output, which uses a sigmoid activation function to produce a match prediction probability. The loss function used was binary cross-entropy loss. After tuning (as seen in Fig. [5]), we found the optimal hyper-parameters to be: run for 3 epochs, $\eta = 0.001$, and $L = 3$ hidden layers.

6 Experimental Design & Results

6.1 Experimental Design

Design Matrix Construction: In both datasets, our input feature vector $\mathbf{x}_i$ represents a single LoL match. For both data sets, we formulate **three design matrices**. Common to all three is the 10 selected champions in a match. Additionally, all three include a measure of player skill. In the professional case, skill is encoded as each professional player's winrate (calculated from the training set). In the casual case, skill is encoded as each player's ELO rating. The first design matrix exclusively uses the features noted above. The second design matrix adds to these the winrates of pairs of champions that are matched against each other. Because we one-hot encode player champion selection, our feature space is large. We therefore prepare a third design matrix using PCA to select the principal components of the second feature design. In all experiments, the label y_i is be the outcome of the match: 1 if blue team won, 0 otherwise.

Choice of Train/Test Data: As previously mentioned, we chronologically order both datasets to avoid encoding future data into our training data. Additionally, since some players played just a handful of matches in the professional dataset, we throw away professional match data where a player plays less than 20 matches. To draw reasonable comparisons between the datasets, we randomly select a subset of $m_c \approx 5000$ from the casual dataset, matching the size of the thresholded professional dataset. Our train/test split was 70/30 due to the fact that the most recent professional data comprised about 30% of the extracted data.

6.2 Results:

Impact of Winrates & PCA: As seen in Fig. [1], the addition of winrates either decreased or did not significantly affect the accuracy in either dataset. However, the inclusion of PCA increased testing accuracy for feature vectors with winrates. From these results we conclude that our models have difficulty accommodating a large feature space which is why including winrates decreases prediction accuracy, and PCA increases prediction accuracy. Moreover, PCA cuts the training time by more than half for SVM and logistic regression with PCA as seen in Fig. [10].

Best Models: In the casual data setting, all models had prediction accuracies within 0.07 of one another, with random forests performing best. In the professional data setting, again all models performed similarly (within 0.07 of one another), though logistic regression had the highest prediction accuracy and the highest F1 score.

False Positives in Professional Data: We observe in the confusion matrices shown in the appendix that our professional models consistently exhibit high false positive rates. We believe this has to do with the blue team's higher win percentage across our professional data set. F1 scores corroborate this conclusion - the F1 scores for the professional data are significantly lower than that of the casual data (.6 vs. .95).

Casual vs. Professional Data Prediction Accuracy: It is most clear that prediction accuracy is much higher in the casual setting. The average inter- and intra- team skill differences shown in table 1 shed some light on why. On average, the winning team's average skill is one point higher

than the losing team's skill. Additionally, on average, the winning team's lane matchup skill is higher. These clear skill differences offer some insight into why, for example, linear classification models perform as well as our nonlinear classification models.

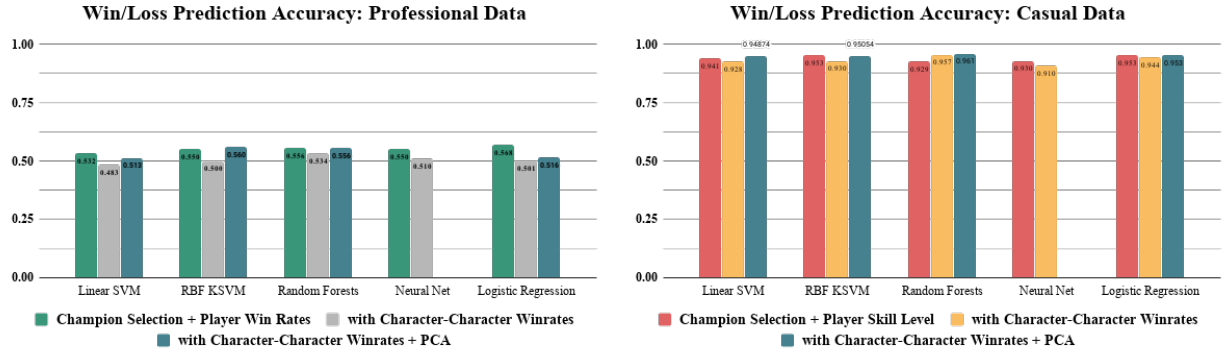


Figure 1: Prediction accuracy comparison between professional and casual data sets. We also compare different feature vectors within each data set.

7 Conclusion & Discussion

We find it surprising that our models perform well on casual data, with prediction accuracies as high as 95%. We are equally surprised that our models do not learn as effectively on professional data, as exhibited by the lower F_1 scores and the highly imbalanced confusion matrices.

While we cannot draw definitive conclusions on the state of game balance in *League of Legends*, we do hypothesize that LoL's matchmaking system might tend to create matches that a specific team is most likely to win. If matchmaking prepared two teams with equal skill level consistently, then an individual may lose several matches in a row if he/she has a 50/50 chance of winning/losing. A player who loses multiple games in a row is not likely to continue playing the game. Alternatively, if the matchmaking system can orchestrate games between two teams that are closely but not exactly evenly matched, then it can ensure that no player loses too many matches in a row.

Acknowledgments

We would like to thank Dr. Shivani Agarwal for providing helpful advice on how to correct some errors in our initial models, as well as clarifying our assumptions about what conclusions can be drawn from our results. We also thank our mentor, Barry Plunkett, who offered helpful insights throughout the execution of our project.

References

[1] Blue wins more than red in league of legends: The stats. https://dotesports.com/general/news/league-of-legends-red-blue-statistics-win-rate-252, 2018.

[2] Logistic regression. https://scikit-learn.org/stable/modules/linear_model.html, 2018.

[3] Playing in random forests in league of legends. https://www.reddit.com/r/leagueoflegends/comments/94bl2r/using_machine_learning_to_predict_game_outcomes/, 2018.

[4] Using machine learning to predict game outcomes at loading screen. https://www.reddit.com/r/leagueoflegends/comments/94bl2r/using_machine_learning_to_predict_game_outcomes/, 2018.

[5] Victoria Hodge, Sam Michael Devlin, Nick Sephton, Florian Oliver Block, Anders Drachen, and Peter Cowling. Win prediction in esports: Mixed-rank match prediction in multi-player online battle arena games. https://www.researchgate.net/publication/321160741_Win_Prediction_in_Esports_Mixed-Rank_Match_Prediction_in_Multi-player_Online_Battle_Arena_Games, 11 2017.

[6] Shayaan Jagtap. How we trained a machine to predict the winning team in league of legends. https://medium.com/trendkite-dev/machine-learning-league-of-legends-victory-predictions-8bc6cbc7754e/, 2018.

[7] Yifan Yang, Tian Qin, and Yu-Heng Lei. Real-time esports match result prediction. https://arxiv.org/abs/1701.03162, 12 2016.

8 Appendix

8.1 Data Set Statistics

	μ	σ
Average Inter-Team Skill Difference (winning - losing)	1.134	2.949
Lane Matchup Skill Difference (winning - losing)	1.241	0.567
Intra-Team Skill Differential (winning)	2.267	1.116
Intra-Team Skill Differential (losing)	2.615	1.241

Table 1: Inter- and intra-skill differences in casual play

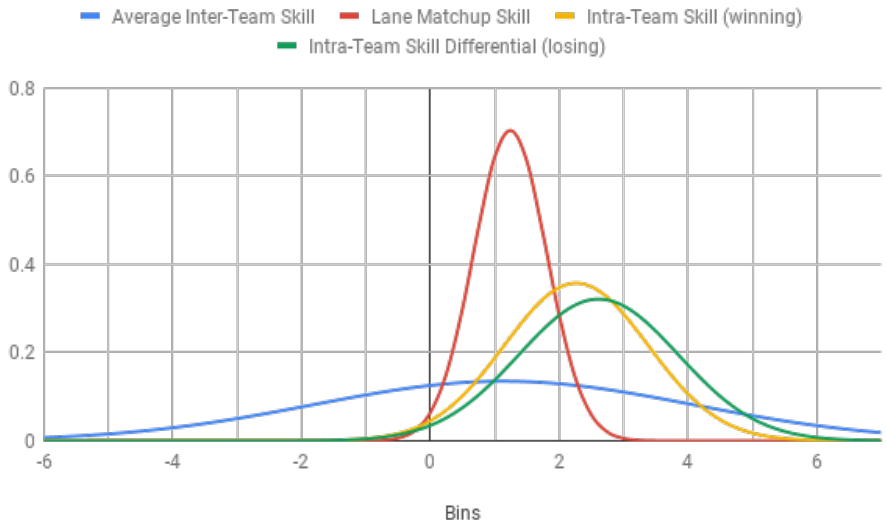


Figure 2: Inter- and intra-skill difference in casual play.

8.2 Tabulated Results

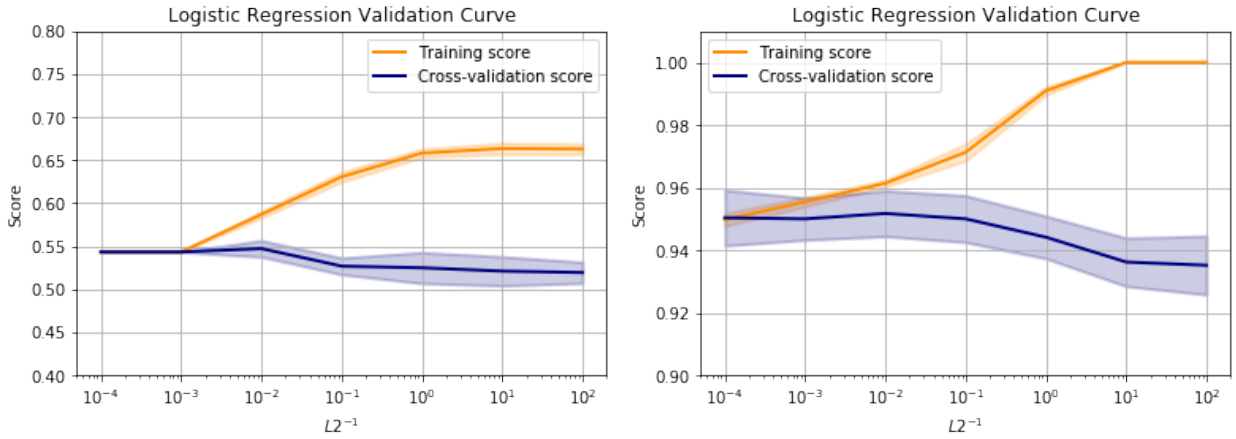
	Professional		Casual	
Model	Prediction Accuracy	F1	Prediction Accuracy	F1
Logistic Regression	0.568	0.720	0.957	0.960
Linear SVM	0.532	0.596	0.941	0.937
RBF SVM	0.550	0.720	0.953	0.950
Random Forest	0.556	0.715	0.957	0.960
Neural Net	0.540	0.660	0.890	0.880

Table 2: Results without champion-champion matchup winrates.

	Professional		Casual	
Model	Prediction Accuracy	F1	Prediction Accuracy	F1
Logistic Regression	0.516	0.55	0.953	0.95
Linear SVM	0.513	0.691	0.949	0.958
RBF SVM	0.560	0.72	0.951	0.959
Random Forest	0.556	0.590	0.961	0.955
Neural Net	0.540	0.59	0.880	0.89

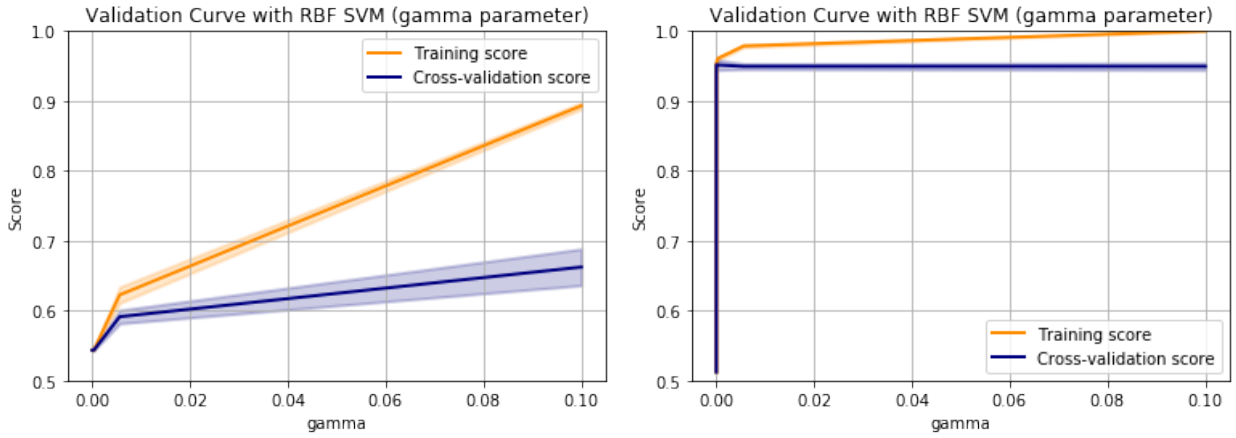
Table 3: Results with champion-champion winrates and PCA.

8.3 Cross-Validation Curves (Logistic Regression, RBF SVM)



(a) Professional (b) Casual

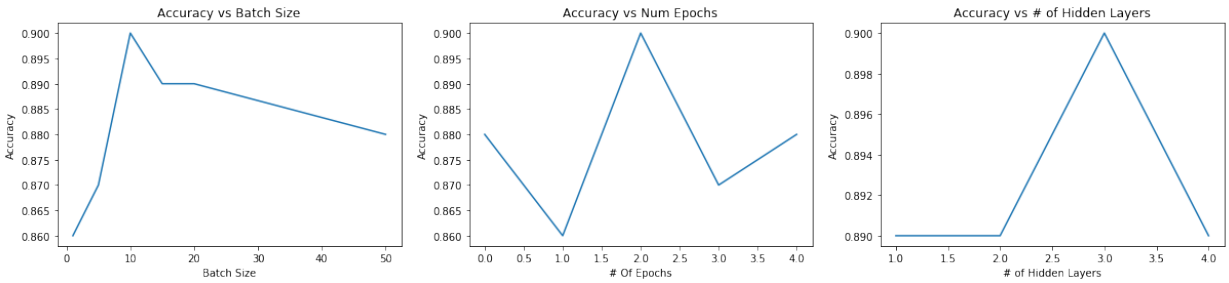
Figure 3: Cross-validation curves for logistic regression.



(a) Professional (b) Casual

Figure 4: Cross-validation curves for RBF SVM.

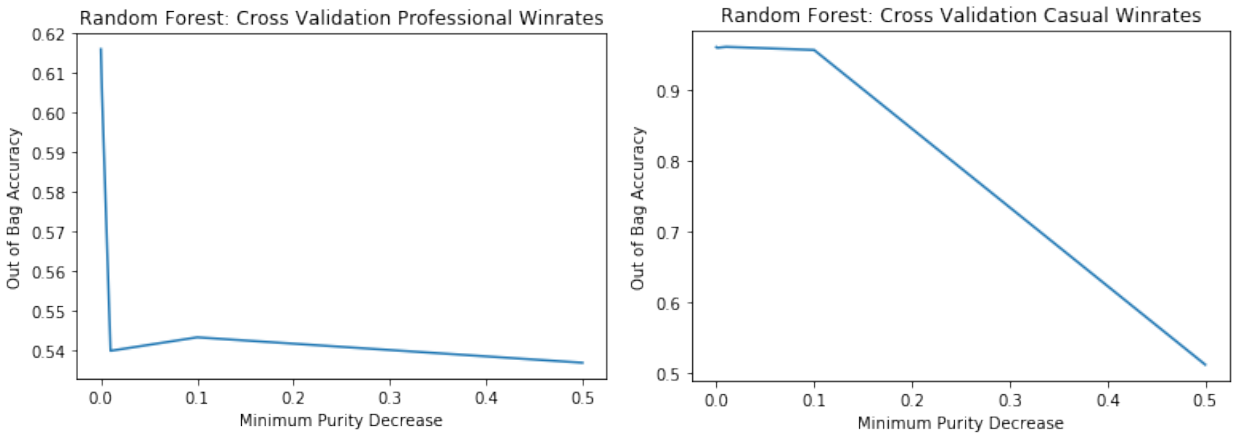
8.4 Neural Network Tuning Curves



(a) Batch Size Tuning (b) # Epochs Tuning (c) Hidden Layers Tuning

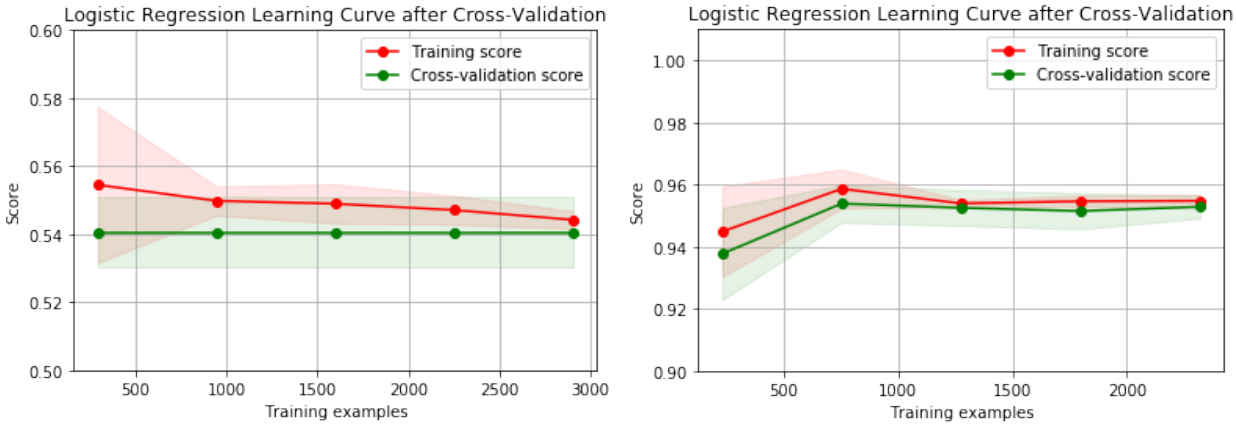
Figure 5: Neural network tuning.

8.5 Random Forest Tuning Curves



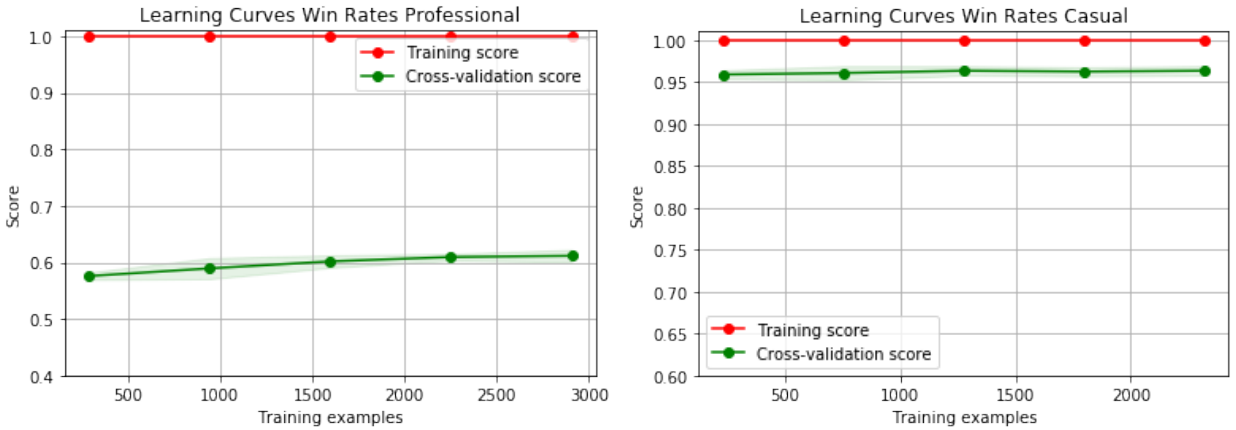
(a) Batch Size Tuning (b) # Epochs Tuning

8.6 Learning Curves



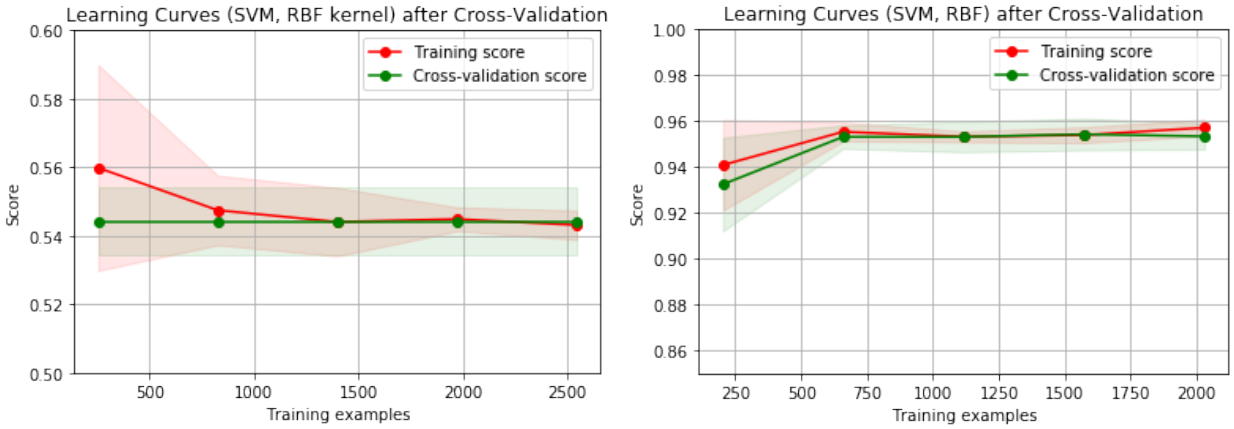
(a) Professional (b) Casual

Figure 7: Learning curves for logistic regression



(a) Professional (b) Casual

Figure 8: Learning curves for random forest



(a) Professional

(b) Casual

Figure 9: Learning curves for RBF SVM.

8.7 Confusion Matrices

		Professional		Casual	
		$\widehat{y}$		$\widehat{y}$	
		-	+	-	+
y	-	33	570	536	29
	+	18	740	24	523

Table 4: Logistic Regression

		Professional		Casual	
		$\widehat{y}$		$\widehat{y}$	
		-	+	-	+
y	-	165	438	543	22
	+	183	575	26	521

Table 5: Random Forest

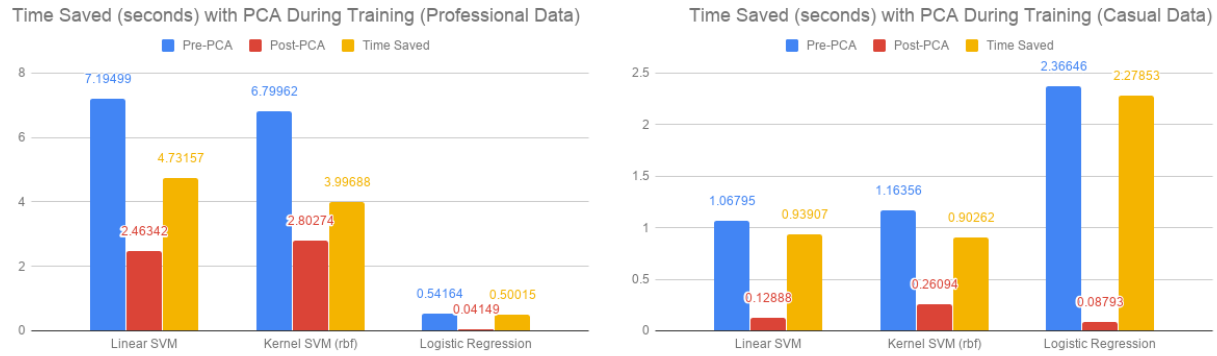
		Professional		Casual	
		$\widehat{y}$		$\widehat{y}$	
		-	+	-	+
y	-	0	603	312	291
	+	0	758	374	384

Table 6: Linear SVM

		Professional		Casual	
		$\widehat{y}$		$\widehat{y}$	
		-	+	-	+
y	-	150	453	529	36
	+	160	597	85	462

Table 7: Neural Net

8.8 Timing



(a) Time saved using PCA on professional data set.

(b) Time saved using PCA on casual data set.

Figure 10: Training time savings using PCA.

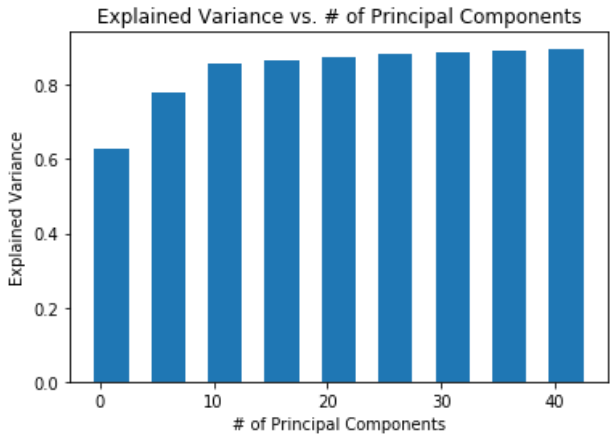


Figure 11: Number of principal components capturing vs. variance captured