**MATH 4323 - Final Report**

**Predicting Tennis Match Outcomes via Statistical Analysis**

Brayan Sosa - 2021182

Fatima Mirza - 2066039

Hatem Saleh - 8006437

Fiza Gubitra - 1985460

**Introduction**:

Our group decided to choose a data set about tennis. The reason we chose this topic is because we all have a passion for sports and this specific data set contains a large number of observations along with both categorical and numerical variables, making it more convenient for us to perform classification task.

The data set came from tennis-data.co/uk and it contains 2222 observations and 36 variables. Here is a short description of each variable.

**ATP:** Tournament number (men)

**Location:** Venue of tournament

**Tournament:** Name of tournament (including sponsor if relevant)

**Date:** Date of match

**Series:** Name of ATP tennis series (Grand Slam, Masters, International or International Gold)

**Court:** Type of court (outdoors or indoors)

**Surface:** Type of surface (clay, hard, carpet or grass)

**Round:** Round of match

**Best of:** Maximum number of sets playable in match

**Winner:** Match winner, Loser: Match loser

**Wrank:** ATP Entry ranking of the match winner as of the start of the tournament

**LRank:** ATP Entry ranking of the match loser as of the start of the tournament

**WPts:** ATP Entry points of the match winner as of the start of the tournament

**LPts:** ATP Entry points of the match loser as of the start of the tournament

**W1 to W5:** Number of games won in 1st to 5th set by match winner

**L1 to L5:** Number of games won in 1st to 5th set by match loser

**Wsets:** Number of sets won by match winner

**Lsets:** Number of sets won by match loser

**Comment:** Comment on the match (Completed, won through retirement of loser, Walkover)

**B365W:** Bet365 odds of match winner

**B365L:** Bet365 odds of match loser

**PSW:** Pinnacles Sports odds of match winner

**PSL:** Pinnacles Sports odds of match loser

**MaxW:** Maximum odds of match winner (as shown by Oddsportal.com)

**MaxL:** Maximum odds of match loser (as shown by Oddsportal.com)

**AvgW:** Average odds of match winner (as shown by Oddsportal.com)

**AvgL:** Average odds of match loser (as shown by Oddsportal.com)

**Research Question:** Will a KNN or SVM model be more accurate for predicting the winner of a tennis match?

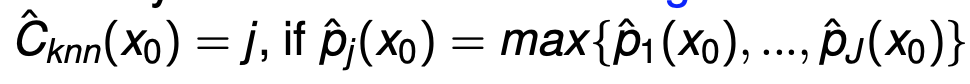
Our goal is to accurately predict which team will win the tennis match . We will be using “results” as our response variable and the remaining relevant data will be used as predictors. We will then compare the data, and analyze if our predicted results match up with the actual outcomes of the tennis matches. This is important because the results of which player wins plays a big role in the betting world. Calculating the statistics on which player will win can help people determine who they should bet on. This will allow people to make the highest profits and bet confidently.

**Methodology**:

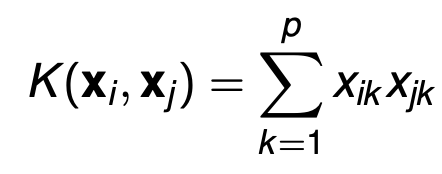
For our models, we implemented K-Nearest Neighbor and Support Vector Machine as our two distinct approaches since we are dealing with supervised learning. For KNN, we used the validation approach. We ran the algorithm for 20 different k values, ranging from k = 1 to k = 20, and compared the error rates for each k value to get a more accurate estimate on which k is best for our model. The KNN classifier decides which class a data point belongs to by using a ‘majority voting principle’. Then using the best K, we could see how accurate our classifications of ‘win’ or ‘loss’ were for each tennis match.

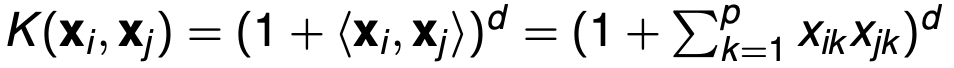
Some things we had to keep an eye out for were underfitting and overfitting. Having a very good training error would indicate overfitting because we may have been fitting the noise rather than the true relationship. On the other hand, we could tell if we were underfitting if our training error was extremely high. KNN has many advantages because it's pretty simple and intuitive, has relatively high accuracy, doesn’t rely on assumptions, and doesn’t need training time. A disadvantage of KNN is that it’s computationally expensive since the algorithm stores all the training data and it’s also sensitive to irrelevant features and the random data split could favor one result over the other.

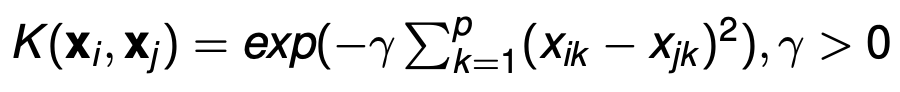
**KNN Formula**: After identifying the K points in the training data closest to x0, we need to estimate the conditional probability for class j as *Pr*(*Y* = *j* | *X* = *x*0) = 1/K *i* ∈*N*0 􏰅 I(*yi* = *j*) . Lastly, we classify the observations to the class with the highest estimated probability.



An advantage of using SVM is that it allows a general framework of enlarging feature space for support vector classifiers efficiently. It also allows some observations to be on the incorrect side of the margin in order to better classify the rest of the training observations. So it doesn’t perfectly separate the two classes. In our case, we used linear kernel, radial kernel, and polynomial kernel and compared the results of each kernel. Linear kernel allows us to linearly separate our dataset into 2 classes using a hyperplane. So in our case, it separates our results into either a win or a loss. Radial and polynomial kernels are useful for when our data is non-linearly separable and they both allow for more flexible boundary shapes. Radial kernels can be more effective in higher dimensional spaces. The biggest advantage of using kernels is that it is more computationally efficient than KNN and can reduce prediction error. A disadvantage of SVM is that it doesn’t give accurate results when there is too much noise, meaning that the classes are overlapping a lot.

**Linear Kernel Formula** → 

**Polynomial Kernel Formula** → 

**Radial Kernel Formula** →

**Data** **Analysis**:

We started our process by first making sure all our variables have values assigned to them. For any variable that was missing a value or was set to N/A, we filled it with a 0 using if statements. After filling in missing values, we found that certain predictors weren’t relevant for our analysis and that they should not be involved in the computations. For example, knowing the location or date of a tennis match won’t impact our predictions of who wins or loses a match. Thus we decided to exclude these types of predictors: Location, Tournament, Date, Series, Court, Surface, Best.of, Comment,Round, and ATP. Doing this reduced the size of our dataset from 36 predictors to 26 predictors. In addition to filtering out the irrelevant predictors, we also created a few new variables: rankDiff (the difference between each winning and losing rank), ptsDiff (the difference between each winning and losing point), set1Diff to set5Diff (the difference between the wins and losses of each set, our data being divided in a total of 5 sets), numSetsDiff, B365Diff, PSDiff, MaxDiff, AvgDiff - setting all of them to numeric values with the as.numeric() function.

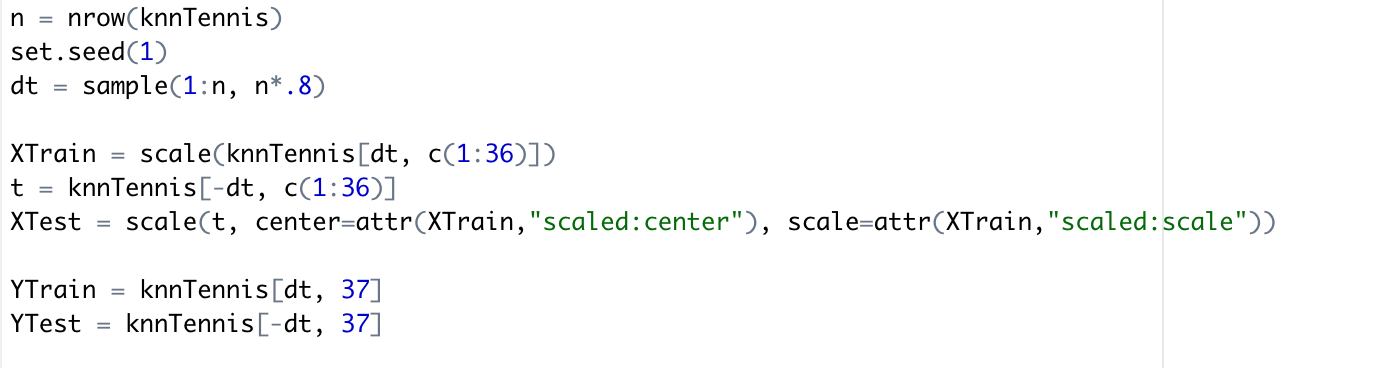
After creating all of our “difference” predictors, we created a binary response variable “result.” In order to do this, we counted the amount of positives and negatives from our “difference” predictors. If the majority of the “difference” predictors were positive, i.e. a win, then “results” was assigned a value of 1. Otherwise, if most of the “difference” predictors were negative or equal to the amount of positives, “results” was assigned a value of 0, i.e. a loss. So a value of 1 means that the player under the “Winner” predictor was predicted to win and they actually won. A value of 0 represents that the player under the “Loser” predictor was predicted to win the match, but they actually lost.

Once we cleaned up our dataset and created our response variable, we began performing KNN. Ignoring the first two predictors, we partitioned our data set so that 80% of our observations could be used for the training set and the remaining 20% could be for the testing set. Then we created XTrain, XTest, YTrain, and YTest and scaled XTrain and XTest. Using our training and testing variables, we began looking for the best K value by implementing the validation set approach. After obtaining the best K, we ran the KNN algorithm and assigned the results to the variable “knnPred”. Lastly, we found the error rate.

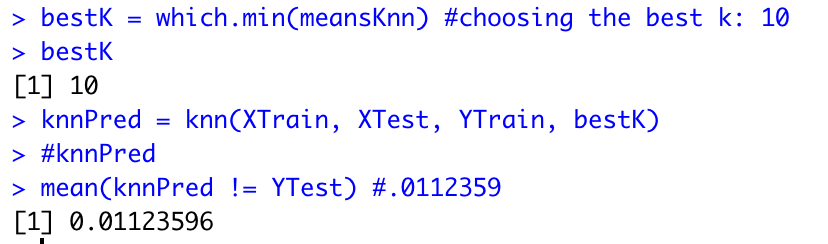
Shortly after performing KNN, we implemented the SVM approach. Once again, we ignored the first two predictors from our current data set and partitioned our data into 80% for training and 20% for testing. We then created our train variable and applied the factor function. We also made sure to tune our data before running SVM each time. We then proceeded to tune our dataset to find the best cost C for linear SVM, the best gamma for radial SVM, and the best degree for polynomial SVM. The cost values that were tested were: 0.001, 0.01, 0.1, 0.5, 1, 5, 10, 100. The gamma values tested were: 0.001, 0.01, 0.05, 0.1, 1, 5, 10. The degree values that were tested ranged from 1-10. After obtaining all the best parameters to perform linear SVM, radial SVM, and polynomial SVM, we proceeded to perform SVM and determined their respective training and testing errors.

**K Nearest Neighbors Algorithm:**

→ Dividing data set into training and testing sets, training model on 80% training data



→ Finding prediction error on 20% testing data

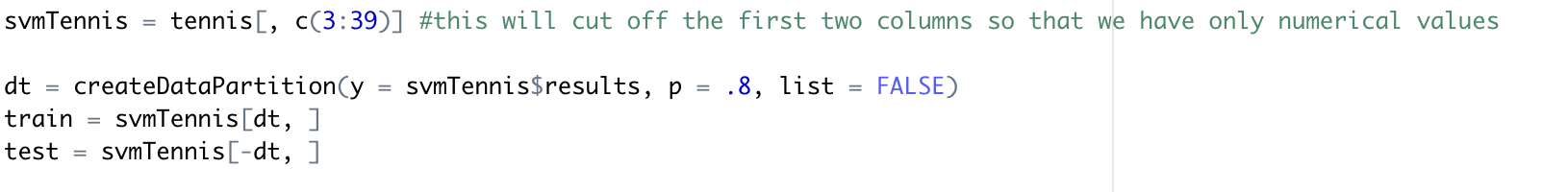
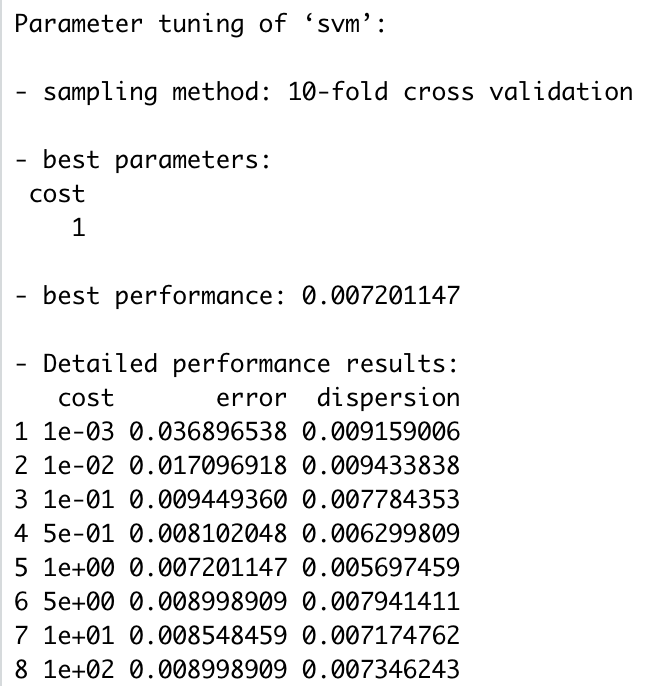
****

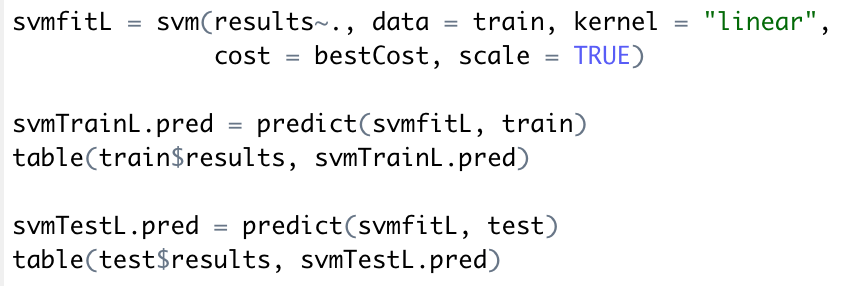
Our optimal K value was 10 and it yielded a test error of 0.01123596, or 1.12%.

**Support Vector Machine Algorithm:**

The main difference between linear, radial, and polynomial SVMs is the way each algorithm creates the separating hyperplane decision boundary between classes. When the dataset is not linearly separable, kernel functions must be used in order to transform the data into a higher dimensional space, allowing us to then perform the algorithms.

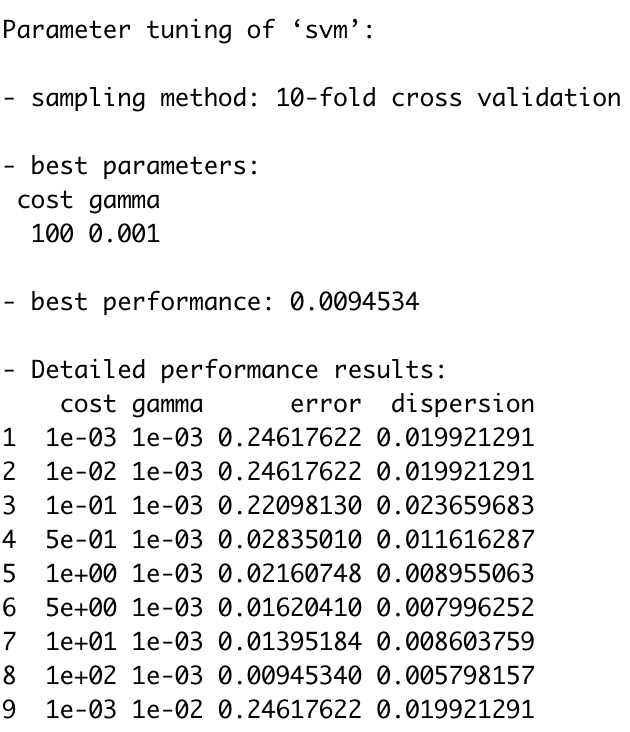
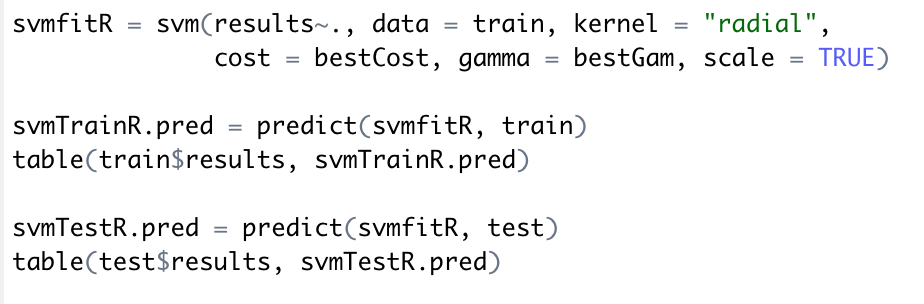
→ Dividing data set into training and testing data

**Linear SVM:**

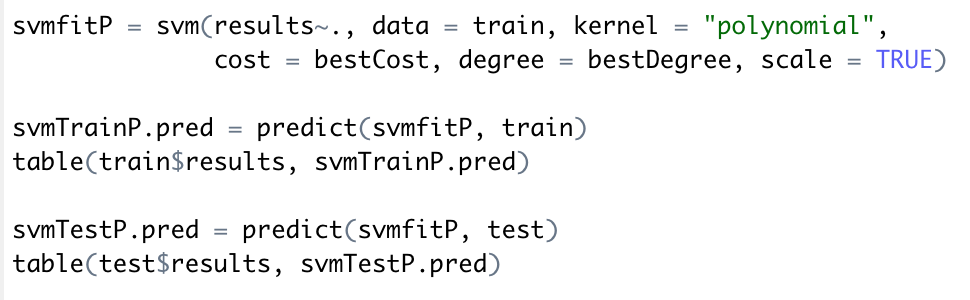
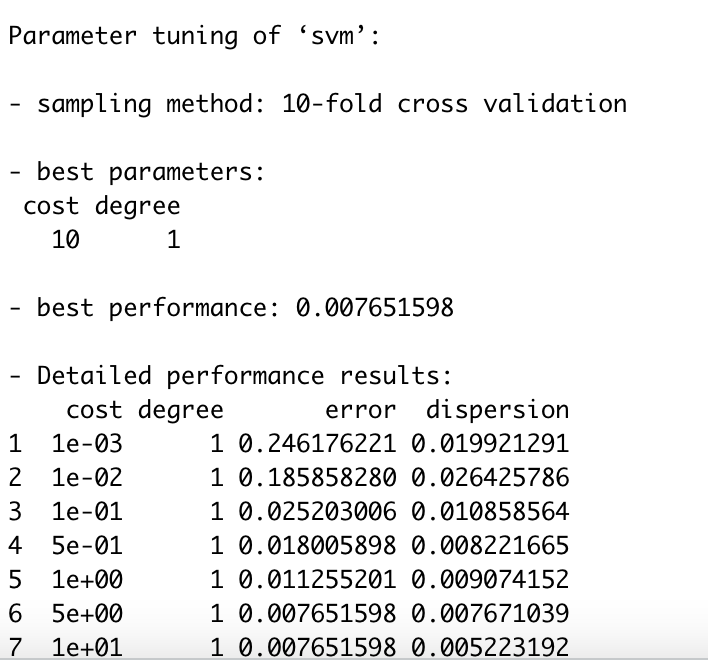
****

The best cost was 1. Using our frequency tables, we calculated our training error to be 0.506% and our test error to be 1.351%.

**Radial SVM:**



Here, the best cost is 100 with a gamma value of 0.001. Using our frequency tables, we calculated our training error to be 0.506% and our test error to be 1.126%

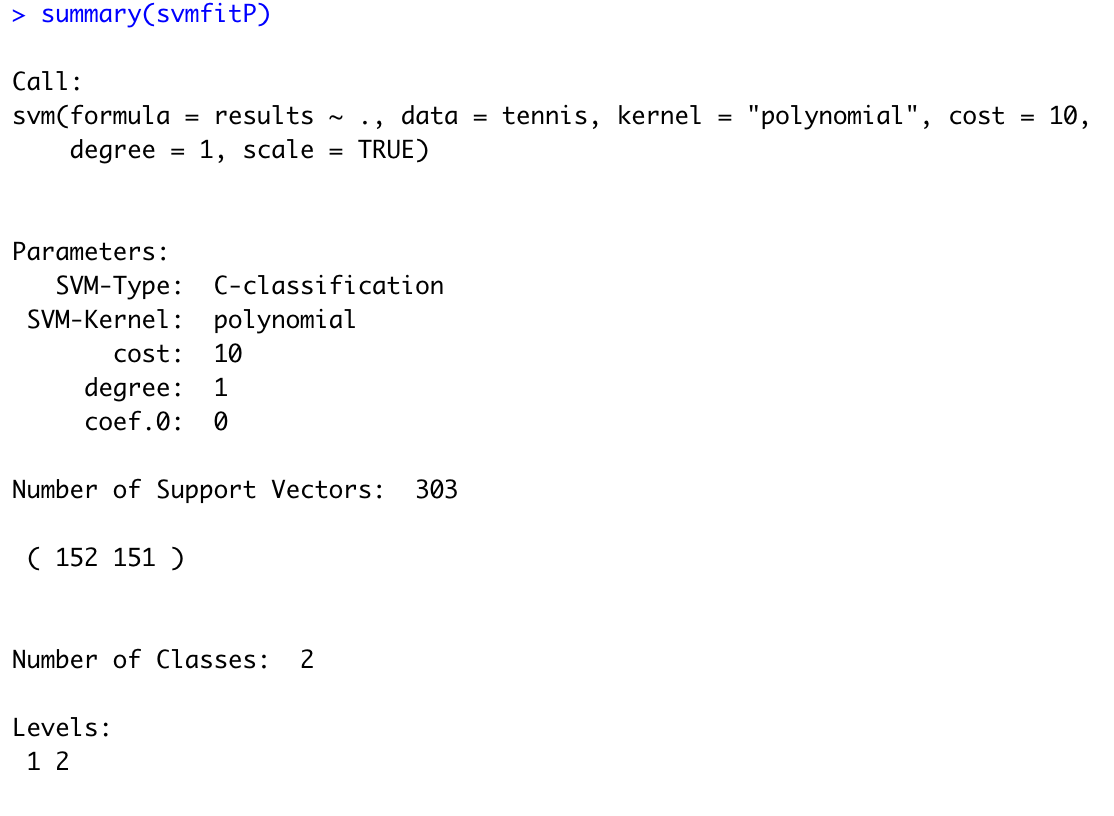
**Polynomial SVM:**

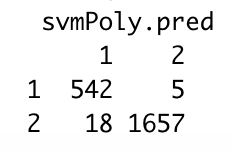
The best cost was 10 and the best degree was 1. Using our frequency tables, we calculated our training error to be 0.731% and our test error to be 0.676%.

**Comparing Results:**

When comparing KNN and SVM, KNN has a significantly higher test error rate so it isn’t optimal. Making a decision between which SVM technique is the most accurate and reliable can be tricky. However, here it can be seen that the testing error as a result of running the linear and radial kernel is significantly higher than the train error rate, suggesting overfitting, i.e., the model cannot be generalized and is fitting the data too close to the training dataset. This eliminates linear kernel and radial kernel as being the most optimal. This leaves us with the polynomial kernel model where we can see the train and test errors have the lowest variance suggesting the best fit. Thus, the most optimal support vector machine kernel is the **polynomial kernel** - with an test error rate of 0.676% - much lower than the linear and radial svm test errors. Even though its training error is slightly higher than the other models’ training errors, we can overlook this since we are more concerned about how well a model can predict the future rather than how well it performs on previous data. Therefore, we choose polynomial svm as the optimal approach.

We then applied polynomial SVM to our entire data set, and our test error was 1.035% Here is the summary of our SVM object along with the frequency table.



****

We can see that our test error went up from 0.607% to 1.035% when we applied polynomial SVM to the entire dataset rather than just 20% of the data. Although this seems like a large increase, it isn’t bad considering that we are working with 2222 observations and that a 1.035 % error rate is still lower than the linear and radial SVM models’ test error rates, 1.351% and 1.126% respectively. We believe we achieved a respectable predictive performance and our results make sense, as shown by our analysis above.

**Conclusion:**

After conducting a thorough analysis of our dataset, we have concluded that the SVM model/approach is more accurate for predicting the winner of a tennis match. Although the SVM model, specifically polynomial SVM, yielded the best results, the KNN model still performed relatively well in predicting the outcome of the tennis matches.

There was a bit of difficulty in performing the data cleaning since we had “duplicate” predictors. For example, we had a rank for both the “winner” and the “loser” predictor and this was the case for a large portion of the data set. This initially made it difficult and puzzling to create a “result” variable for our data set, since it felt as though we were working with the same variable twice. Another difficulty that we encountered was choosing the best model for our data set. KNN did have the best test error rate compared to all the models but since we had no real way of obtaining a training error rate, we had to take into consideration external factors such as the computational demand for each model to reach our conclusion.

Something we can do to improve our data analysis is increase the number of subsets used for our SVM algorithm. This will help us yield more accurate results, improving the efficiency of our analysis. Another possible change we could make is to split our data set differently. So instead of having a 80- 20 split for our training and test data, we could do a 70-30 split. Lastly, we could also cleanse our data before implementing each algorithm to ensure our results are centered around the most relevant data, and void of any noise.

**References**:

Barla, N. (2022, November 14). *Overfitting vs underfitting in machine learning: Everything you*

*need to know*. neptune.ai. Retrieved December 8, 2022, from https://neptune.ai/blog/

overfitting-vs-underfitting-in-machine-learning

Hamilton, D., Pacheco, R., Myers, B., & Peltzer, B. (1970, January 1). *KNN vs. SVM: A*

*comparison of algorithms*. US Forest Service Research and Development.

Retrieved December 8, 2022, from https://www.fs.usda.gov/research/treesearch/62328

Oleszak, M. (2021, September 21). *SVM kernels: What do they actually do?* Medium. Retrieved

December 8, 2022, from https://towardsdatascience.com/svm-kernels-what-do-they-

actually-do-56ce36f4f7b8

**Appendix:**

1. Frequency tables for SVM models → used to calculate training and test errors

| Model Type | Freq. Table for Training Set | Freq. Table for Testing Set |
| --- | --- | --- |
| Linear |  |  |
| Radial |  |  |
| Polynomial |  |  |