****

School of Computing, Engineering and Built Environment

Machine learning and Data analysis

Coursework 1

Word count: 1549

*Author: Jonathan Ward*

*Student Number: S1902178*

Contents

[Description of data and the problem 3](#_Toc124113439)

[Data Pre-processing 5](#_Toc124113440)

[Constructing and tuning the SVM Model 7](#_Toc124113441)

[Testing Results 10](#_Toc124113442)

[Discussion 11](#_Toc124113443)

[Appendices 12](#_Toc124113444)

[Main Python Script 12](#_Toc124113445)

[References 15](#_Toc124113446)

[Figure 1 - First look at chosen Dataset 3](#_Toc124113370)

[Figure 2 - Pre-processing where number of instances and columns are displayed 3](#_Toc124113371)

[Figure 3 - Data Types of each variable in the dataset 5](#_Toc124113372)

[Figure 4 - Code in script responsible for removing unwanted columns 6](#_Toc124113373)

[Figure 5 - Proof there is no duplicate rows in the dataset 6](#_Toc124113374)

[Figure 6 - Double checking there are no null values in dataset 7](#_Toc124113375)

[Figure 7 - Checking balance of how many games were won and lost in the dataset 7](#_Toc124113376)

[Figure 8 - Target variable defined for classification of dataset 7](#_Toc124113377)

[Figure 9 - Required variables for classification of dataset 8](#_Toc124113378)

[Figure 10 - First successful SVM model created 8](#_Toc124113379)

[Figure 11 - GridSearchCV tuned model 9](#_Toc124113380)

[Figure 12 - Results of GridSearchCV tuning 9](#_Toc124113381)

[Figure 13 - Classification of validation dataset using hyperparameters found during tuning the model 10](#_Toc124113382)

[Figure 14 - Classification of testing dataset 11](#_Toc124113383)

# Description of data and the problem

The chosen data structure contains information on in game statistics from the video game league of legends, consisting of information about how well players performed up until the 10-minute mark of a game. Each instance of the data structure counts as an individual game, with almost 10 thousands games in the dataset, containing almost 40 different columns all containing statistics on how well the two teams playing against each other in each game performed.

Graphical user interface, application, table, Excel

Description automatically generated

Figure 1 - First look at chosen Dataset

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 2 - Pre-processing where number of instances and columns are displayed

Player statistics by the 10-minute mark of each of these instances is collected, with the dataset’s variables containing:

* Kills
* First bloods
* Gold difference between teams
* Experience point difference between teams
* Deaths
* Assists
* Objectives
* Minions killed
* Etc

These are all very important pieces of information which all contribute overall to the outcome of a game of league of legends. These pieces of information can decide whether or not one team or the other team ends up winning a game of league of legends.

As outlined in figures 1 & 2, there are 9,879 instances with 40 variables in this dataset, entirely made up of numerical values without any null or empty values in the dataset.

The report problem is as follows: Can the chosen dataset be used to predict the outcome of a game of league of legends based on player statistics collected in the first 10 minutes of a game. Given the nature of the question and dataset, this can be considered a classification problem, meaning a support vector machine (SVM) algorithm would be suitable to solve the problem, which is a effective supervised machine learning algorithm. Supervised machine learning algorithms involve labelled data, with these algorithms being very good at predicting new data based on old data provided. (McGregor, 2020)

# Data Pre-processing

Before any model can be created, the data must be properly sanitised and processed, as this can ensure that the data isn’t volatile or unstructured in anyway that could negatively affect the SVM classifier.

Checking the dtypes of the dataset, we can see that all data type are either an integer or a float, which is fine for an SVM model and does not need changed. Usually if there are any objects or strings present, they would need sanitised before it is used by the SVM classifier.

Text

Description automatically generated

Figure 3 - Data Types of each variable in the dataset

Next any variables that are not needed for the classifier are removed, as data about the blue team and red team is available, however data about the red team is not necessary, as the aim of the model is to predict if data collected can predict a win or a loss for an individual team. This cleans up the dataset and reduces the amount of processing when it comes to constructing and tuning the SVM model.

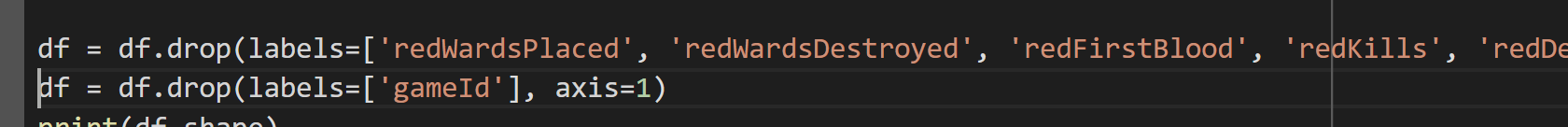


Figure 4 - Code in script responsible for removing unwanted columns

Next the dataset must be checked for any null values and duplicate values, which there are none of in the dataset, so no processing is necessary at this point.

The number of instances in each variable are counted as well.

A picture containing text

Description automatically generated

Figure 5 - Proof there is no duplicate rows in the dataset

Then if there are any duplicates present, which there are not, they are dropped. The data is also checked for any null values to ensure there aren’t any values that could affect the accuracy of the model.

Text

Description automatically generated

Figure 6 - Double checking there are no null values in dataset

The value of ‘blueWins’ was double checked to ensure there are an even number of instances where the blue team won and game and lost a game. Since this will be the target variable for the model this helps the accuracy of the model.

Graphical user interface, text, application

Description automatically generated

Figure 7 - Checking balance of how many games were won and lost in the dataset

# Constructing and tuning the SVM Model

Once the dataset has been properly sanitised and pre-processed the model can be constructed. The SVM model was chosen due to its ability to create a classification model between two groups of data, creating a hyperplane between these two groups to be able to predict data. In this case the first group is “blueWins” which specifies whether the game was won or lost. The rest of the data specific to how well the team is performing in the first 10 minutes of the game is used to create a hyperplane between the models target data, “blueWins” and the rest of the dataset.

Text

Description automatically generated

Figure 8 - Target variable defined for classification of dataset

First two variables called “X” and “y” should be specified, with X being the value of “blueWins”, and y being the rest of the dataset. X represents the target variable required to build a SVM classifier, and the y represents all other columns of the dataset.

Text

Description automatically generated

Figure 9 - Required variables for classification of dataset

The dataset is then split up into three parts, with the training set holding 70% of the dataset, with the remaining 30% of the dataset being split between the validation and test set.

Next the SVM classifier is created using the “fit” function, with the training set passed through as the parameter for the model.

After this the model has been created, the “predict” function can be used to measure and print out the accuracy of the model we have created.

Text

Description automatically generated

Figure 10 - First successful SVM model created

The accuracy of the first SVM model created was 73%. This model should be tuned appropriately to see if the accuracy of the model could potentially be greater, as the current model has not been tuned at all.

For tuning the model, a GridSearchCV function was used to see if a more accurate model could be found, while also finding appropriate parameters and hyperparameters which can be used to create more accurate models using the validation and testing datasets which have not been classified yet.

Hyperparameters are important in determining how a model is trained and change the final version of the model. (Stainsbury, 2023)

This function takes in parameters and is used to brute force through several SVM classifications to find the most accurate. However, this process is very time and performance sensitive, with this process taking as much as an hour to complete using the provided parameters.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 11 - GridSearchCV tuned model

Text

Description automatically generated

Figure 12 - Results of GridSearchCV tuning

The accuracy produced from the tuned model was actually smaller than the first model created, however this process provided the correct hyperparameters which can be used in a model which uses the validation and testing dataset which was split up before any SVM model was constructed.

# Testing Results

One a tuned model has been created, the hyperparameters taken from the results of the GridSearchCV function can be used alongside the validation dataset which can be properly classified.

Text

Description automatically generated

Figure 13 - Classification of validation dataset using hyperparameters found during tuning the model

This model produces an accuracy of 75%, the most accurate model produced so far.

Finally, the classification of the testing dataset is outputted, showing an accuracy of 73% which is slightly lower than the classification of the validation data set.

Graphical user interface, text, application

Description automatically generated

Figure 14 - Classification of testing dataset

# Discussion

Before any analysis or manipulation of data can even begin to start it must be properly pre-processed. Pre-Processing is responsible for keeping a dataset precise and valid. (Indeed, 2022)

In the case of the chosen dataset, very minimal processing was needed to get the most out of the dataset. Some columns were removed to keep the volume of data lower when constructing the model. Data was collected for the enemy team during a game of league of legends however the data that was chosen as the target variable had nothing to do with whether the red team won or not and was the only necessary pro-processing required to carry out the SVM classification of the dataset.

The chosen dataset, very luckily, was in a perfect condition to be used in a SVM classification, with only numerical values, and didn’t have any empty or null values, across almost 10 thousand different instances in the dataset. This saved quite a lot of time before the model was created, however consideration toward pre-processing that should usually happen were acknowledged in the report and python script.

Once the data had been processed properly setting up a SVM classification was quite simple, as the accuracy already was sitting at 72%, however after using GridSearchCV as a tuning method for the data set, unfortunately the tuning only resulted in a decimal increase, with the accuracy ending up being 73%, however this process allowed the correct hyperparameters for classification of the validation data set to be the most accurate model produced, being 75% accurate.

The tuning of the model could’ve have resulted in a better accuracy, however finding the correct hyperparameters ended up increasing the average by around 4%.

Overall, 75% is a successful accuracy for the model, and proves that the dataset has been successfully classified through the SVM algorithm and would not have been achieved without proper pre-processing and tuning of the dataset.

# Appendices

## Main Python Script

*# import libraries*

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

from sklearn.metrics import accuracy\_score, precision\_score

from sklearn.preprocessing import MinMaxScaler, StandardScaler

from sklearn.svm import SVC

*# Link the script to the dataset*

df = pd.read\_csv("data/high\_diamond\_ranked\_10min.csv")

df.head()

*# Drop columns which aren't needed for the model to work properly*

df = df.drop(labels=['redWardsPlaced', 'redWardsDestroyed', 'redFirstBlood', 'redKills', 'redDeaths', 'redAssists', 'redEliteMonsters', 'redDragons', 'redHeralds', 'redTowersDestroyed', 'redTotalGold', 'redAvgLevel', 'redTotalExperience', 'redTotalMinionsKilled', 'redTotalJungleMinionsKilled', 'redGoldDiff', 'redExperienceDiff', 'redCSPerMin', 'redGoldPerMin'], axis=1)

df = df.drop(labels=['gameId'], axis=1)

print(df.shape)

*# Show top 20 instances of the dataset*

df.head(20)

*# check for any duplicate rows in the dataset, and if so remove them*

duplicate\_rows\_df = df[df.duplicated()]

print ("Duplicate rows: ", duplicate\_rows\_df.shape)

df.count()

df = df.drop\_duplicates()

print(df.count())

print(df.isnull().sum())

*# Check there are even amount of instances where blue team loses but also wins their games*

print(df['blueWins'].value\_counts())

*# Specify that the variable 'blueWins' is the target varaible of the SVM model, while the rest of the dataset is represented by another variable*

X = df.drop(['blueWins'], axis = 1)

y = df['blueWins']

print(X.head())

print(y[0:5])

*# create variables used in the classification, specifying how much of the dataset is split into 3 individual datasets which are used for indiidual models*

X\_train, X\_tmp, y\_train, y\_tmp = train\_test\_split(X, y, test\_size=0.3, random\_state=1)

print("Size of training X:", X\_train.shape)

X\_validation, X\_test, y\_validation, y\_test = train\_test\_split(X\_tmp, y\_tmp, test\_size=0.5, random\_state=1)

print("Size of the validation X:", X\_validation.shape)

print("Size of the test X:", X\_test.shape)

*# First SVM classification is created, with its accuracy printed out*

clf1 = SVC()

clf1 = clf1.fit(X\_train, y\_train)

y\_pred1 = clf1.predict(X\_validation)

print("Accuracy:", metrics.accuracy\_score(y\_validation, y\_pred1))

*# GridSearchCV tuning to perfect the previously made model to try increase its accuracy*

*# This also will find any hyperparameters which can be used for testing models using the validation and testing datasets previously declared*

from sklearn.model\_selection import GridSearchCV

parameters = [{'kernel': ['linear'], 'gamma':[1, 0.1, 0.01], 'C': [1, 10, 100, 1000]}]

clf2 = SVC()

grid = GridSearchCV(clf2, parameters, cv=5, scoring='accuracy', verbose=10)

grid.fit(X\_train, y\_train)

print('Best Hyper-parameters: ', grid.best\_params\_)

print('Accuracy: ', grid.best\_score\_)

*# Hyperparameters found from GridSearchCV tuning used to increase accuracy of classification of validation data set*

clf3 = SVC(kernel = 'linear', C=10)

clf3 = clf3.fit(X\_train, y\_train)

y\_pred3 = clf3.predict(X\_validation)

print("Accuracy:", metrics.accuracy\_score(y\_validation, y\_pred3))

*# Classification of testing data set*

y\_pred = clf3.predict(X\_test)

print("Accuracy:", metrics.accuracy\_score(y\_test, y\_pred))

# References

Indeed., 2022. What is data pre-processing? (with importance and examples) Available at : <https://ca.indeed.com/career-advice/career-development/data-preprocessing#:~:text=Importance%20of%20data%20preprocessing&text=It%20improves%20accuracy%20and%20reliability,It%20makes%20data%20consistent>.

McGregor, M., 2020. SVM Machine learning Tutorial – What is the support vector machine algorithm, Explained with code examples. Available at: <https://www.freecodecamp.org/news/svm-machine-learning-tutorial-what-is-the-support-vector-machine-algorithm-explained-with-code-examples/>

Stainsbury, M., 2023. Model Tuning. Available at: <https://www.mlexam.com/model-tuning/>