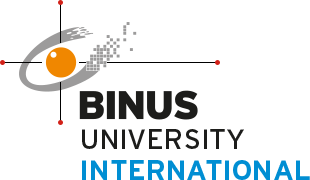
**DATA SCIENCE**

**FINAL PROJECT REPORT**



**Project Title:**

Drafter

**Group Members:**

Jeremy Ponto (2301891525)

Kevin Herman Otnieliem (2301891550)

1. **Problem Introduction and Hypothesis**

Dota 2 is an online multiplayer game developed by Valve. It is a multiplayer online battle arena (MOBA) based game. Dota 2 is a game in which two teams of five players compete against each other, each defending and occupying their own base on the map. Each of the ten players is in charge of a powerful character known as a "hero," who all have different skills and play styles. During a match, players gather experience points and gear for their heroes in order to beat the heroes of the other team in player versus player combat. The first team to destroy the opposing side's "Ancient," a massive structure housed within their base, will be declared as winners.

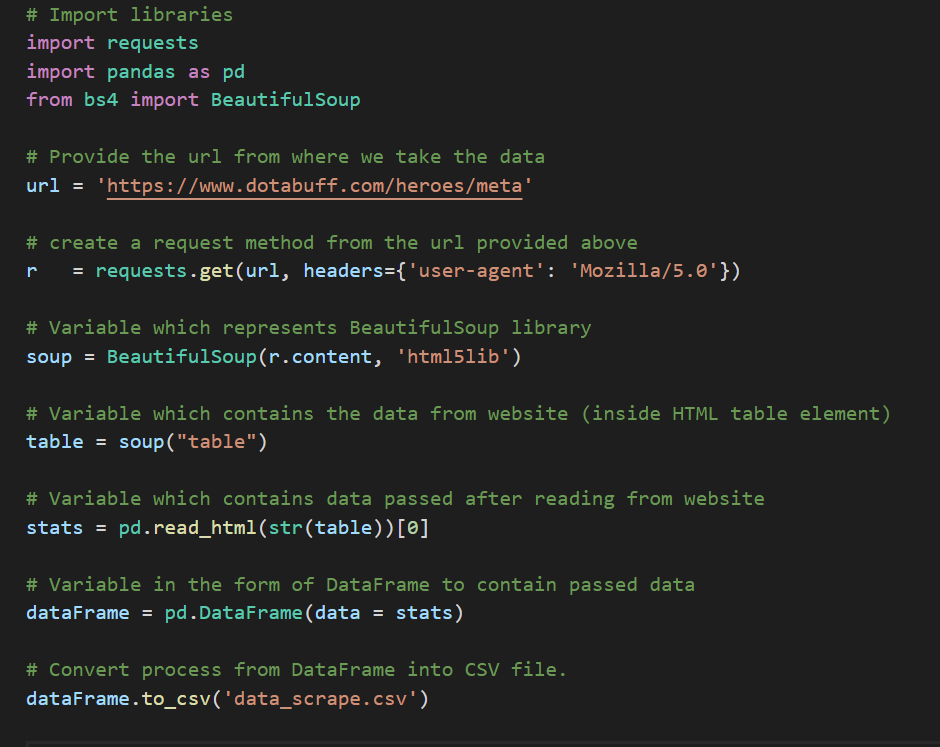
Dota 2 has a thriving esports culture, with teams from all around the world competing in leagues and tournaments. The Dota Pro Circuit is a series of competitions managed by Valve that give qualification points for direct invitations to The International, the game's main annual tournament. Dota 2 is the most lucrative esports game because of a crowdfunded prize money structure that has seen amounts as high as US$40 million. In Dota 2 tournaments, there is a moment called drafting phase. This process is where each team picks and selects their hero to play. Most tournaments are covered by a team of on-site analysts who provide commentary and analysis for the ongoing matches, similar to how traditional sporting events are covered. Usually they are called panelists. Panelists' jobs is to analyze and give a brief explanation about the teams and their drafting phase. Sometimes the panelists' predictions are not correct. So, we are trying to create a project to help on predicting hero drafts in Dota 2 tournaments. This will help the panelist to at least give a correct prediction so that the audience can understand why certain heroes are picked. Because predictions for drafting in Dota 2 are very crucial for teams to win.

We found that there is an existing application that works the same as ours called Shiny App. This software has a great feature that allows you to choose the hero when your teammates are ready. They are using straightforward algorithm. In a game, they presume that the total Gold and Experience per minute for the entire team are fixed, which is correct if both teams are playing aggressive or protective. If one of the team members picks four supporting heroes who don't require a lot of gold or experience, the last person has a lot of room to "harvest" the gold. So picking one hero from one of the most diverse "cores" could be a good idea. But what if their teammates have already used up all of the gold in the room? That implies that the member needs to pick a support hero who requires few resources and aids the "core" in gaining control of the game. Here is the documentation of the existing project about picking Dota 2 hero: <https://nycdatascience.com/blog/student-works/r-visualization/dota-2-heroes-items-selection/>

Looking from the background and the problem we are going to solve, we have an objective which is to suggest the best possible hero role, specifically for the Safe Lane position because this position is the most crucial part of one’s team.

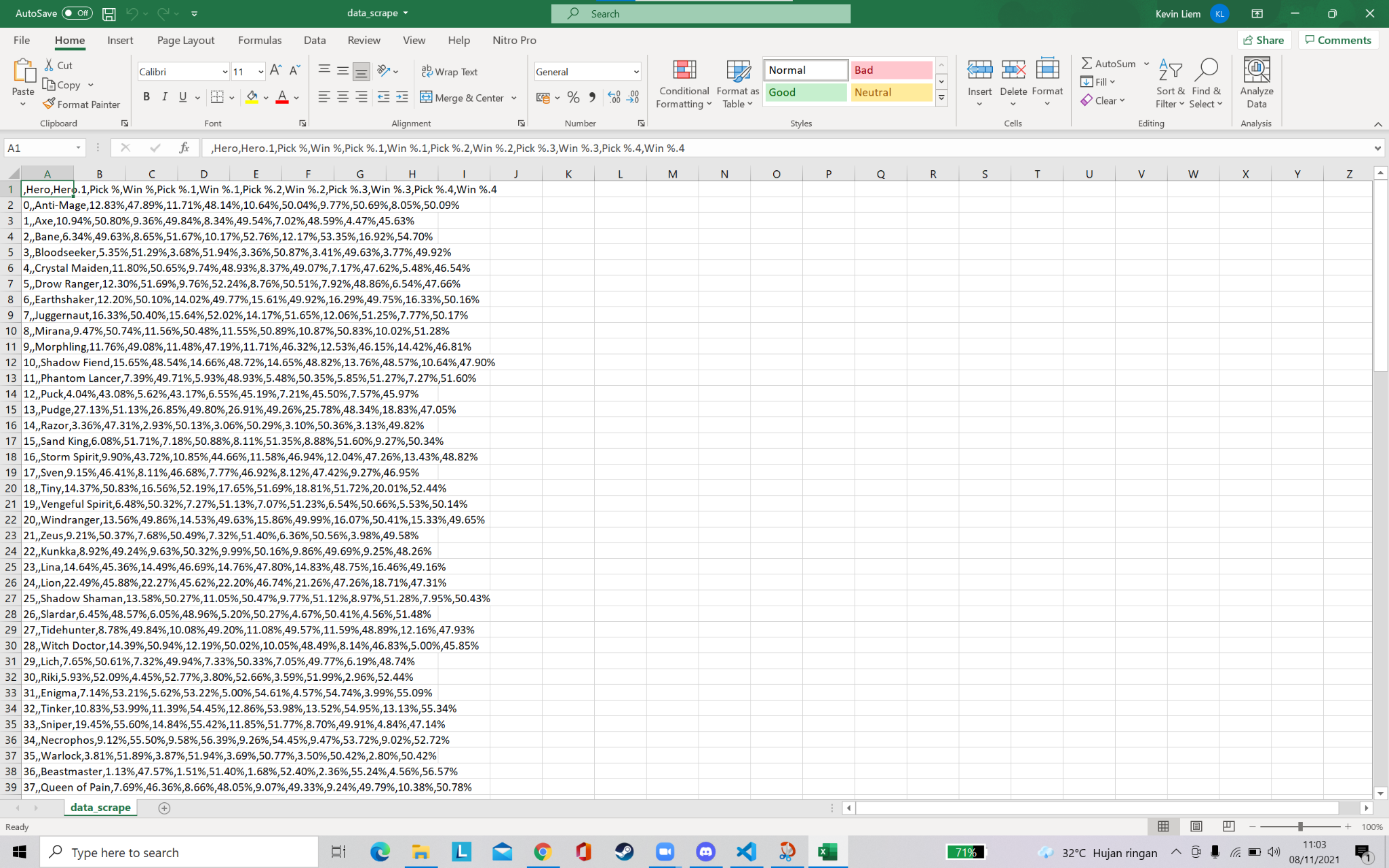
1. **Dataset**

For the Dataset we obtain data via dotabuff.com website. The web contains data for pick and win rate for each meta hero at the moment. To obtain the data, we are making use of Python's web scraping. We use a library called BeautifulSoup to scrape data from the website. And to transform the data frame into a csv file, we are making use of the pandas library. Here is how we obtain the data.



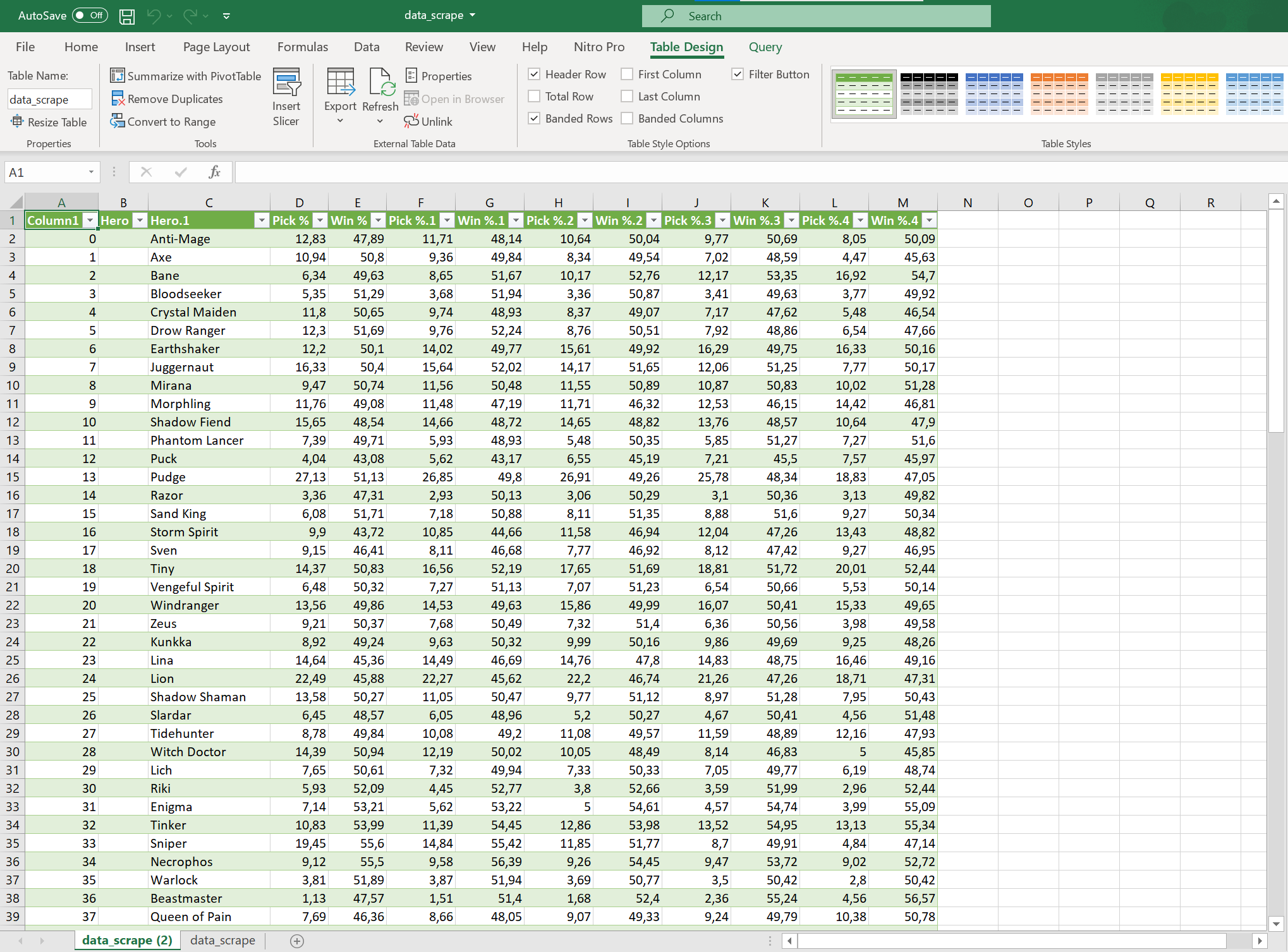
**Figure 2.1** Web scraping code snippet.

Here is the data we obtained.



**Figure 2.2** Original Data obtained from web scraping.

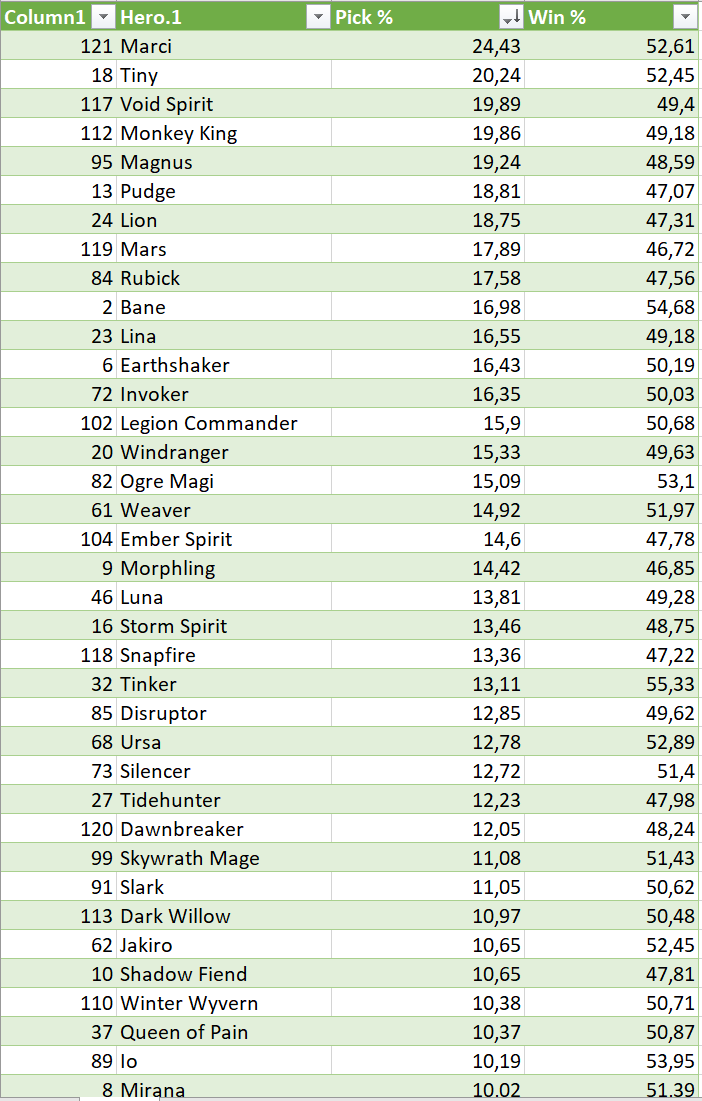
The csv file will seem very messy and hard to work on. So, by using excel we try to make it more readable and understandable.



**Figure 2.3** Neat Data obtained from web scraping.

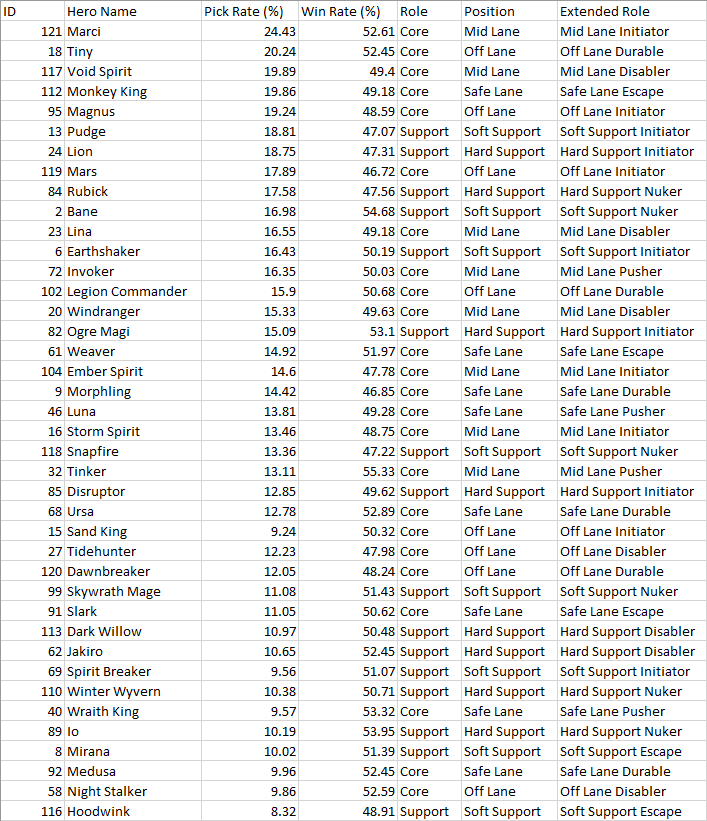
1. **Data Preparation and Processing**

Now we have a dataset for the hero and their pick and win rates. But we will only use the last two columns because those columns represent the top tier level (professional) players’ statistics. We will not use the rest of the columns, so we delete them. So, here is what our data set will look like.



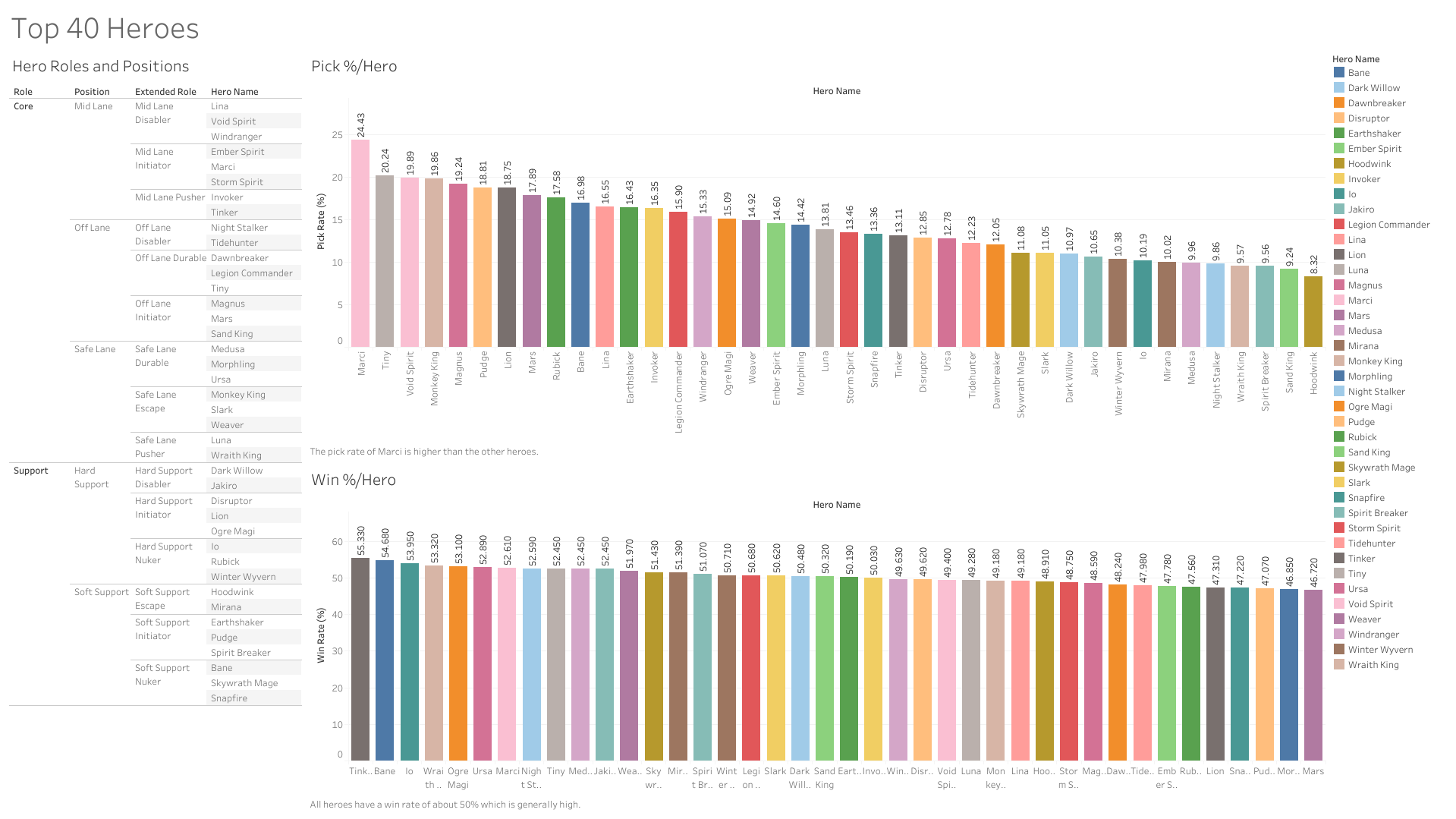
**Figure 3.1** Pick rates and win rates of all heroes.

But for us to make the best draft predictions, those features shown above are not enough. And we also consider that Dota 2 has 121 heroes. In which, not all of those heroes are frequently used in the meta right now. So, we decided to take only the top 40 heroes with the highest pick rate. And we also add some features to consider when picking certain heroes, such as positions, and roles.



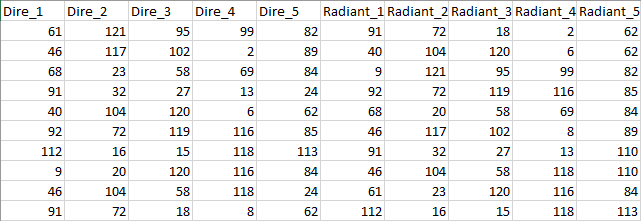
**Figure 3.2** Pick rates, win rates, roles, and positions of the top 40 heroes.

To visualize the data above, we need to connect the csv file to Tableau so we can make bar graphs for hero pick and win rates as well as the roles and positions of these top 40 heroes (link: <https://public.tableau.com/views/DSFinalProject_16363750910480/InterestingFindings?:language=en-US&publish=yes&:display_count=n&:origin=viz_share_link>).



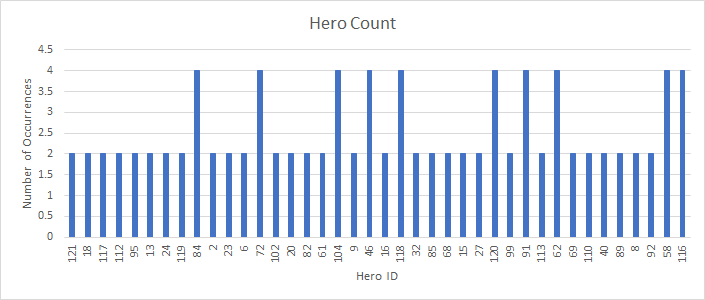
**Figure 3.3** Bar graphs for pick rates and win rates and a table for roles and positions of the top 40 heroes.

From the top 40 heroes we have selected from the dataset obtained from dotabuff.com, we make 10 hero lineups so that our model can predict one of the columns which is a position from a team which will be asked by the program.



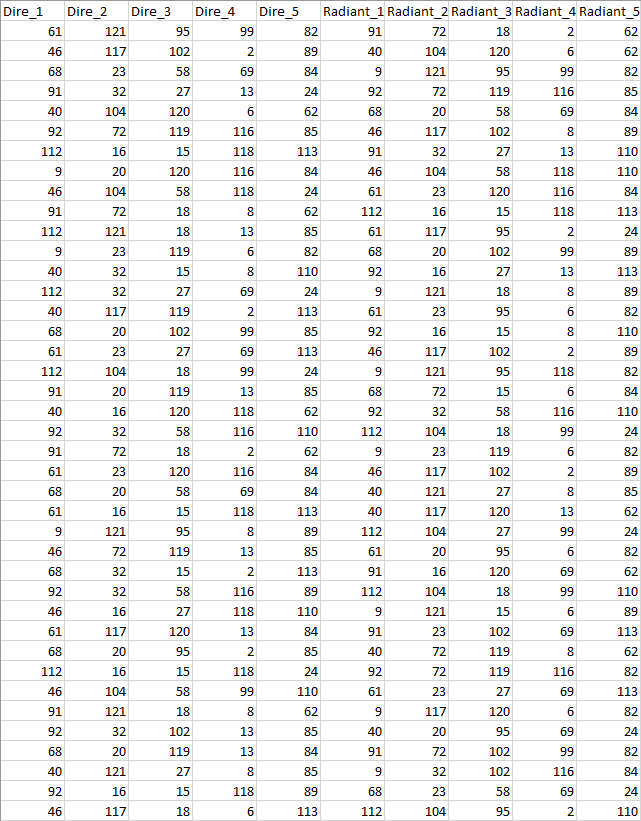
**Figure 3.4** 10 Rows of hero lineups in the form of hero IDs.

To see whether the dataset is balanced or not, we plot a histogram to count the number of occurrences of all heroes. From the figure below, we can imply that the dataset is imbalanced. There are several heroes which have 4 counts.



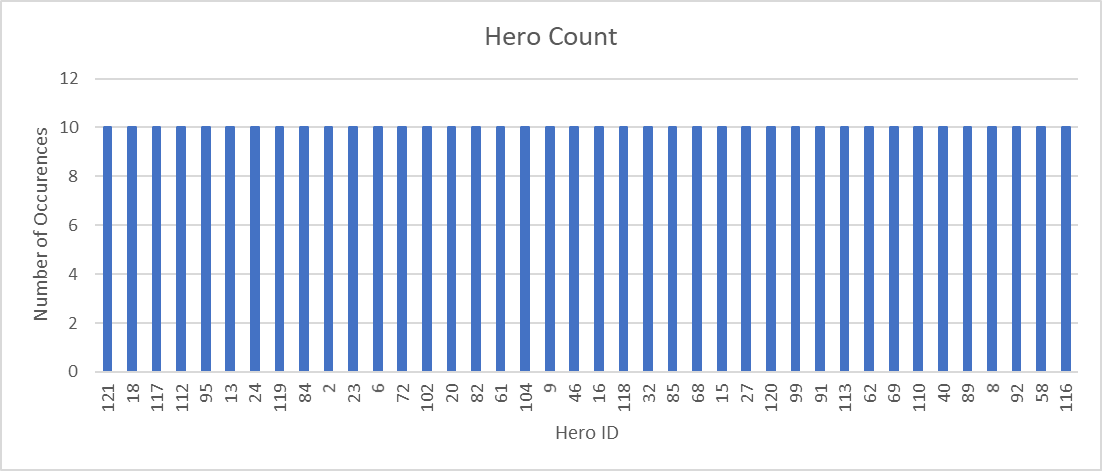
**Figure 3.5** Count of all heroes from the hero lineups.

To deal with the imbalance, we will add 30 rows to the dataset so that it will be balanced.



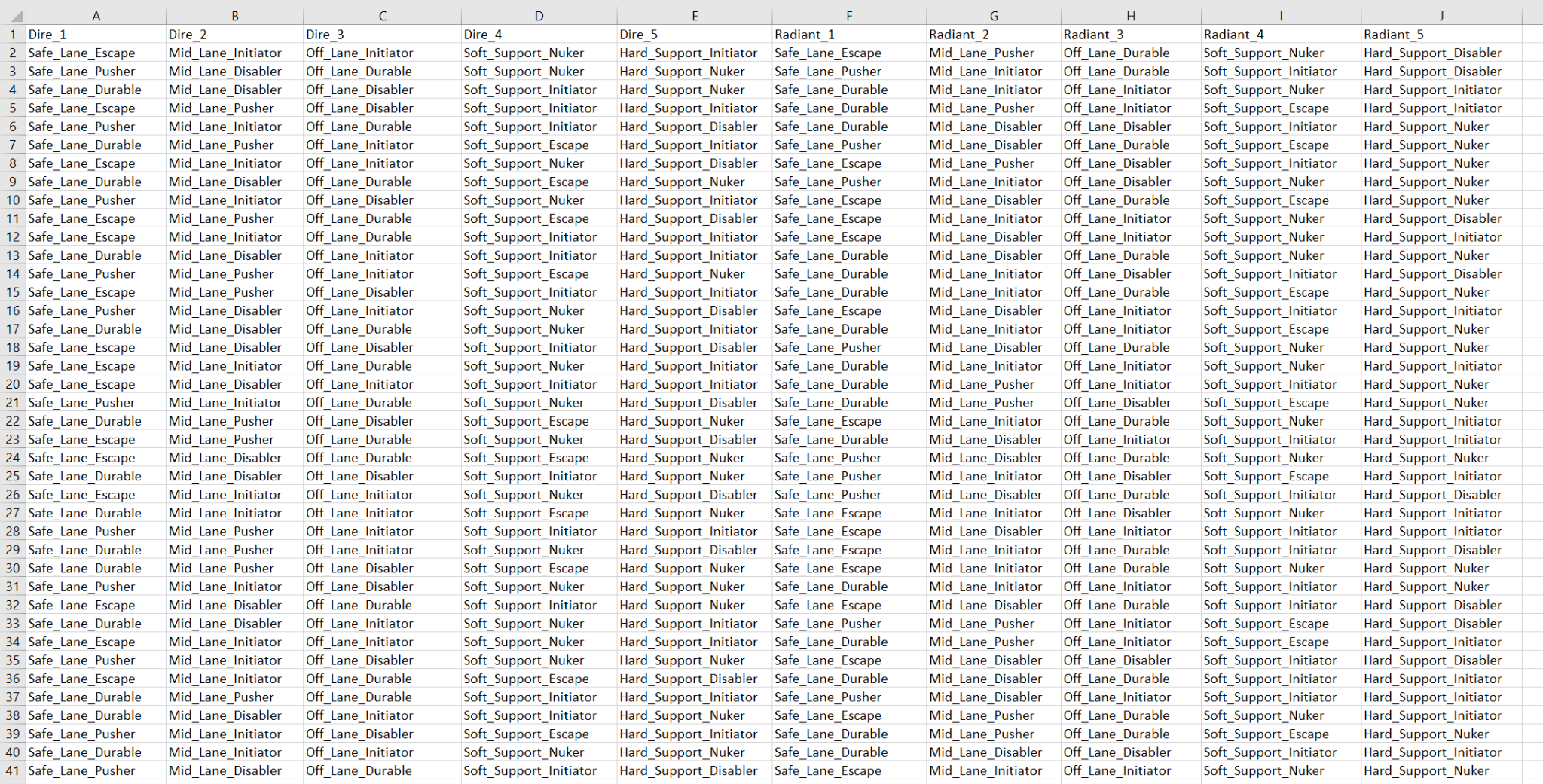
**Figure 3.6** 40 Rows of hero lineups in the form of hero IDs.

To see whether the dataset is balanced or not after an additional 30 rows, we plot a histogram to count the number of occurrences of all heroes. From the figure below, we can imply that the dataset is balanced. All heroes have 10 counts.



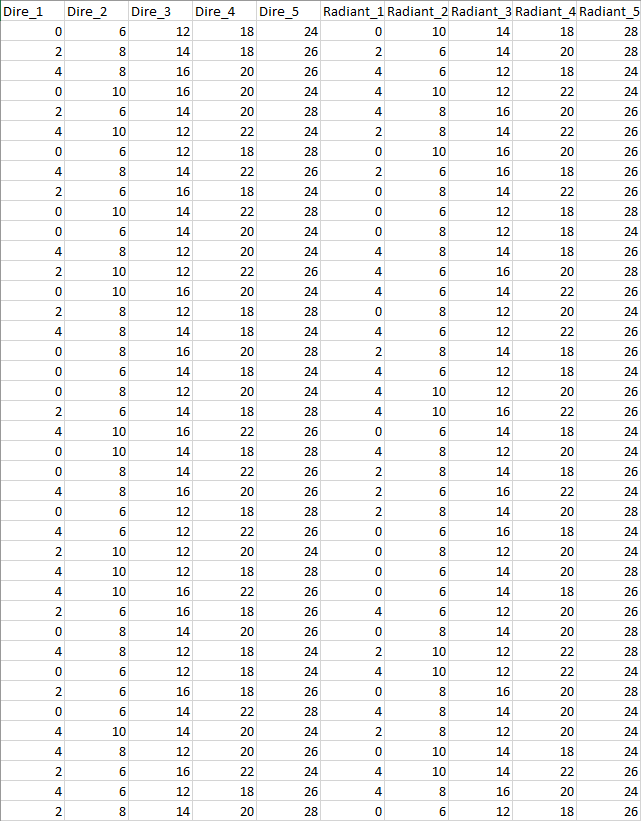
**Figure 3.7** Count of all heroes from the hero lineups after an additional 30 rows.

Due to limited dataset rows, we replaced the hero IDs with the extended roles of heroes.



**Figure 3.8** 40 Rows of hero lineups in the form of hero extended roles.

This dataset will be used by the model to predict the extended roles of the Dire\_1 column. In order to make the model easier to predict we are going to alias the extended roles into numbers, as shown by the screenshot below.

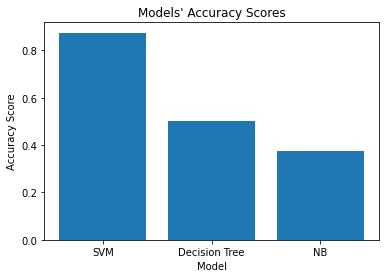


**Figure 3.8** 40 Rows of hero lineups in the form of hero extended roles that have been aliased.

1. **Model and Techniques**

There are several possible models which we propose for predicting the result according to the features, such as Naive Bayes (NB), Decision Tree, and Support Vector Machines (SVM). To decide on which technique to use, we need to know beforehand what is the recommended problem to be used with these models. The classification technique is recommended when the label is discrete whereas the regression technique is recommended when the label is continuous. By looking at our dataset, we have 10 columns, namely Dire\_1, Dire\_2, Dire\_3, Dire\_4, Dire\_5, Radiant\_1, Radiant\_2, Radiant\_3, Radiant\_4, and Radiant\_5. Because Safe Lane is the most crucial part of one team’s line up in Dota 2, the label will be Dire\_1 and the rest will be the features. We know that the label and feature is discrete, so we decided to implement the classification technique.

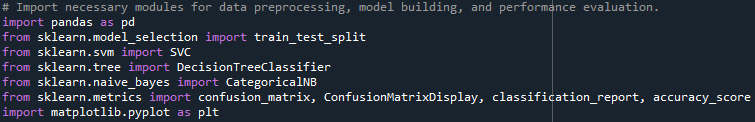
We prefer using the SVM model over other models. The reason for the preference is because SVM gives the best accuracy score followed by Decision Tree Model and Naive Bayes comes last. Below is the screenshot of the accuracy score differences.



**Figure 4.1** Visualization of accuracy scores between three models.

We use Pandas, Scikit-Learn, and Matplotlib as our tools. The reason why we use these tools is because we need to extract the data from a csv file, preprocess and split them into training and test data, build several models and evaluate them as well as create and show visualizations for the evaluations done for the models.

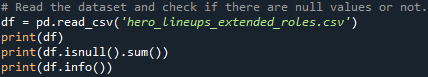
These are the tools used to do the classification and prediction task.



**Figure 4.2** Tools that are used in classification and prediction.

Pandas is used to extract the data and preprocess them. The train\_test\_split function from sklearn module is used to split the dataset into train and test data. The SVC object from the sklearn.svm module is used to build a new SVM classifier model. The DecisionTreeClassifier object from the sklearn.svm module is used to build a new Decision Tree classifier model. The CategoricalNB object from the sklearn.naive\_bayes module is used to build a new NB classifier model. We chose CategoricalNB over other types of NB models because by looking at our dataset, we have categorical values which are discrete. The confusion\_matrix function from the sklearn.metrics module is used to see the amount of predicted labels respective to the true labels. The ConfusionMatrixDisplay object from the sklearn.metrics module is used to visualize the confusion matrix generated from the confusion\_matrix function. The classification\_report function from the sklearn.metrics module is used to show the precision, recall, F1 score, accuracy, macro average, and weighted average of the models. The accuracy\_score function from the sklearn.metrics module is used to show the accuracy of the models. The matplotlib.pyplot module, aliased as plt, is used to make a bar graph to compare the accuracy scores of the models and show all visualizations, including the confusion matrix of the models.

We start by reading the csv file as a dataframe. We need to show the data frame to ensure that we read the correct csv file. We need to check for null values in the dataframe so that we can make sure if the dataset is clean. We need to show the data type of each column to verify whether we have used numeric values as labels for the extended roles which we are going to predict.



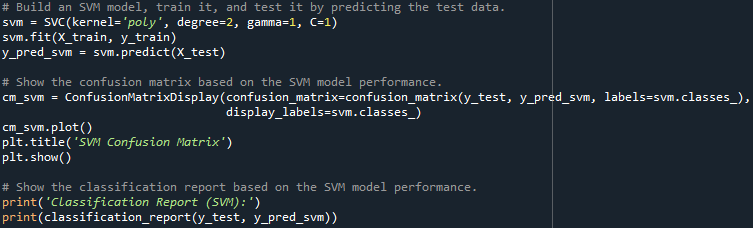
**Figure 4.3** Screenshot of the preprocessing data process.

After making sure that we have a clean dataset, we need to split our dataset into training data and test data.



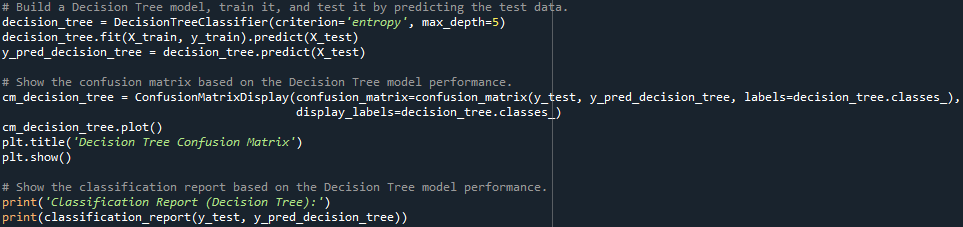
**Figure 4.4** Screenshot of the splitting process of the dataset.

After splitting the dataset, we try to use the SVM Classifier as our model to make the prediction. Then, to evaluate the result of the prediction we use the evaluation method tools from sklearn.metrics, which are confusion matrix display and classification report tools. From these tools, we obtain the accuracy score for the SVM Classifier model.



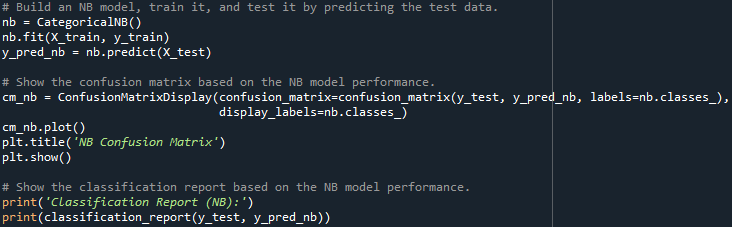
**Figure 4.5** Screenshot of the SVM model prediction process and evaluation method.

To compare with the SVM model, we try to use the Decision Tree Classifier as another model to make the prediction. Then, to evaluate the result of the prediction we use the evaluation method tools from sklearn.metrics, which are confusion matrix display and classification report tools. From these tools, we obtain the accuracy score for the Decision Tree Classifier model.



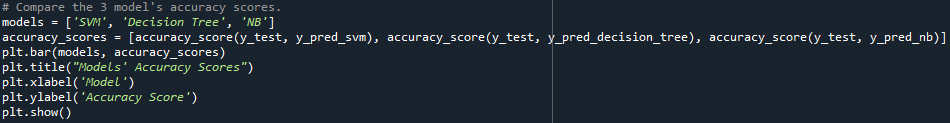
**Figure 4.6** Screenshot of the Decision Tree Classifier model prediction process and evaluation method.

Another model to compare is Naive Bayes Classifier, we try to use it as another model/technique to make the prediction. Same as previous models, to evaluate the result of the prediction we use the evaluation method tools from sklearn.metrics, which are confusion matrix display and classification report tools. From these tools, we obtain the accuracy score for the Naive Bayes classifier model.



**Figure 4.7** Screenshot of the Naive Bayes model prediction process and evaluation method.

Last but not least, we compare them by visualizing it using pyplot. The result of this visualization is shown in Figure 4.1 of this section.



**Figure 4.8** Screenshot of the visualization process to display the accuracy score differences of each model.

1. **Evaluation Method**

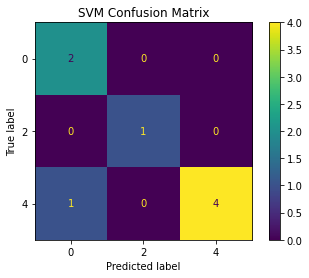
In the model we split the dataset into 80 percent training data and 20 percent testing data (as shown in figure 4.4).

Precision is a classifier's ability to avoid labeling a negative occurrence as positive. It is defined for each class as the ratio of true positives to the sum of true positives and false positives. A classifier's recall is its capacity to discover all positive occurrences. It is defined for each class as the ratio of true positives to the sum of true positives and false negatives. The F1 score is a weighted harmonic mean of precision and recall, with 1.0 being the highest and 0.0 becoming the lowest. The number of actual instances of the class in the provided dataset is referred to as support.

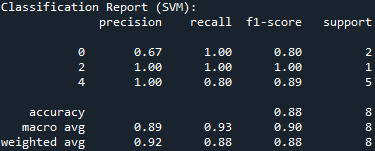
The reported averages include macro average (averaging the unweighted mean per label), weighted average (averaging the support-weighted mean per label), and sample average (only for multilabel classification). Micro average (averaging the total true positives, false negatives and false positives) is only shown for multi-label or multi-class with a subset of classes, because it corresponds to accuracy otherwise and would be the same for all metrics. All of these are used as our evaluation technique.

1. **Results and Discussion**

We make use of several models to test which is suitable for our dataset (as shown in the previous part). We decided to use the SVM classifier for our project. From the explanation in the previous section we now understand the reason for using SVM. Below we provide the classification report and confusion matrix for the SVM model.



**Figure 6.1** Confusion matrix for the SVM model.



**Figure 6.2** Classification report for the SVM model.

We can refer to the screenshot above that SVM model provides a high number of accuracy score which is 0.88 or 88%.

1. **Conclusion and Recommendation**

After looking at the results and discussion, it is very surprising that the model can achieve a high accuracy with only 40 rows of data from the dataset. Therefore, the objective which is to suggest the best possible hero role, specifically for the Safe Lane position because this position is the most crucial part of one’s team has been accomplished.

We are fully aware that our dataset is less than enough. We believe that the more the dataset, the smarter the model will be at predicting the class values. In addition to that, experimenting with more models, such as a neural network can be taken into consideration because of its superior flexibility when it comes to hyperparameter tuning and it may be compared in several ways with the existing model we have made. Perhaps, these recommendations can be implemented in the future.

1. **Source Code**

<https://github.com/jeremyponto/Data_Science_Final_Project>

1. **Contact Details**

Jeremy Ponto

Email : [jeremyponto01@gmail.com](mailto:jeremyponto01@gmail.com)

Mobile Phone : 081252627270

Kevin Herman Otnieliem

Email : [kevinherman547@yahoo.com](mailto:kevinherman547@yahoo.com)

Mobile Phone : 087834235403

1. **Questions and Answers**