Horse Racing Machine Learning Model

Data Science: Big Data Management System & Tools

Group 17

With Spark MLlib and Datbricks community
Data Source: Big Data Derby
Published code: Group project 17 - Databricks
Exploratort Analysis Dashaboard: Group project 17 - Dashboard

Horacio Lovo

Natalia Restrepo

Table of content

Introduction

This report seeks to analyze horse racing tactics, drafting strategies, and path efficiency to illustrate the likelihood of winning a derby race, based on significant race variables, track conditions, and in particular the odds that are currently being placed on each individual horse/jockey that has participated in a race. The dataset is provided by The New York Racing Association (NYRA) and the New York Thoroughbred Horsemen's Association (NYTHA) which conduct world-class thoroughbred racing at Aqueduct Racetrack, Belmont Park, and Saratoga Race Course.

Objective

To develop a supervised learning classification algorithm to predict the likelihood of winning a derby race, given a set of observations.

This analysis will utilize various techniques and methods learned from the Big Data Management Systems and Tools course, to ensure big data handling and high accuracy with our predictive model.

Background

Horse racing in New York has a long and distinguished history, starting with America's first recognized racing event in 1665 at the Newmarket course in Salisbury, New York. Today, the NYRA oversees and operates the pre-eminent thoroughbred horse racing circuit in North America. Races are conducted at three historic venues: Aqueduct Race Track, Belmont Park, and Saratoga Race Course (NYRA, 2022).

- Aqueduct Race Track is a 1-1/8 mile oval track located in New York City. It operates primarily from fall to spring each year.
- Belmont Park, at 1-1/2 miles, is the largest track in the United States and is located on Long Island in New York. It is home to the third jewel of racing's Triple Crown, the Belmont Stakes.
- Saratoga Race Course is a 1-1/8 mile track in Saratoga Springs, New York. Established in 1863, it is one of the oldest, continually run major sports venues in the United States.

Given the costs and complexity of horse racing, statistical models capable of better understanding and valuing horses, jockeys, and strategies can greatly benefit owners and team members by providing insights into their horses and their chances of winning a race.

Dataset

The Big Data Derby dataset from Kaggle was utilized, which included extensive details from nearly every race conducted in the venues Aqueduct Race Track, Belmont Park, and Saratoga Race Course during 2019. The dataset provided by NYTHA and NYRA includes 4 .csv files with relevant information about tracks races took place, race date, weight carried, jockey, odds, course type, course condition, etc. with around 5,228,430 entries.

Moreover, we added two extra datasets that complement the Big Data Derby database: The Equine Death and Breakdown, and Big Data Derby Global Horse IDs and places datasets. These datasets contain information about the horses (name and ID) and information about the horses that broke down, were

injured, or died at New York State race tracks, which we considered relevant for our purpose. Table 1 presents a summary of the different files used in this project, refer to Appendix 1 and Appendix 2 for more information about the variables within each dataset.

Table 1. Big Data Derby datasets

File	Description
nyra_start_table.csv	Horse/jockey race data.
nyra_race_table.csv	Racetrack race data.
nyra_tracking_table.csv	Tracking data.
nyra_2019_complete.csv	Combined table of the three above files.
equine.csv	Report lists horses that have broken down, been injured, or have died at New York State race tracks
horse_ids.csv	The provided dataset lists horses in races.
horse_names.csv	The name of the horses.

Methodology

Considering the size of our files, we opted to use the platform Databricks Community and pyspark libraries to run our algorithms.

Dataset cleaning and preparation

Data Cleaning

We started by importing and looking at the schema of the seven .csv files described in Table 1. From this analysis, we observed that tables *nyra_start_table*, *nyra_race_table*, and *nyra_tracking_table* shared the following features: track_id, race_date, and race_number, whereas table *nyra_start_table* and *nyra_tracking_table* shared feature program_number. We also observed that *nyra_start_table* and *nyra_2019_complete* did not contain the first row with the name of the columns. Lastly, we observed that the values for all the variables in the seven files were of string type. We then created a schema for each file and re-assigned the correct type and column name for each variable. Finally, we checked for missing and null values in all the data frames and continued to prepare the final dataset.

Data Preparation

First, we created a new column in dataset *nyra_2019_complete* called horse_ID, where we added together the information contained in variables track_id, race_date, race_number, and program_number for each row. This column was created with the purpose of joining the *nyra_2019_complete* dataset with *equine, horse_ids,* and *horse_names* datasets altogether later for the final data frame.

For the dataset *equine*, we started filtering the data frame to only contain the races information that took place in 2019 in New York, then we dropped all the variables that we considered were not needed and kept only columns: incident_date, incident_type, track, horse, and jockey_driver. Finally, we created a new column called days_from_injury to calculate the number of days a horse was rested from racing after having an injury.

Lastly, we made sure that the jockey names and the track names for each race matched in all data frames to avoid any duplicate information that could lead to a wrong fit in our model, and joined all the datasets altogether producing the data frame df_prepped which we used to train and test our model.

Exploratory Analysis

From the data exploration done in the data frame df_prepped, we observed the following observations:

Most of the races performed in New York in 2019 took place in track Aqueduct, while only 21.5% of races took place in Saratoga (see Figure 1). The most common course type was Dirt with a track condition of Fast (see Figure 2). The condition of the course depends on the course type and the weather before the race. Heavy rains can make the track soft while sunny days can make the track firm. When a dirt track is dry, it is labeled as fast, referring to a fast-dry track, which can actually produce slightly slower times. Only a few tracks were labeled as good, these are the most desirable tracks where the sand is still drying out from rainfall, producing faster times, but is not wet enough that is considered sloppy or muddy. Moreover, Claiming Dirt races were the most common type of race, followed by Maiden Special Weight races and Maiden Claiming (see Figure 3).

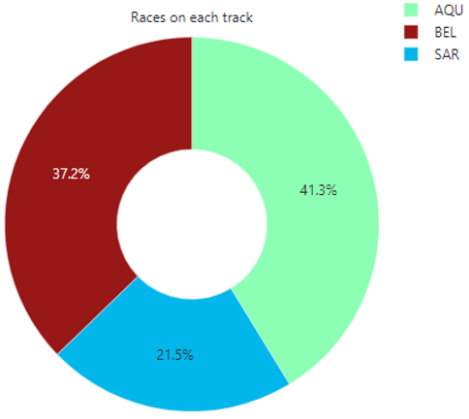

Figure 1. Percentage of races that took place in three different tracks in New York during 2019

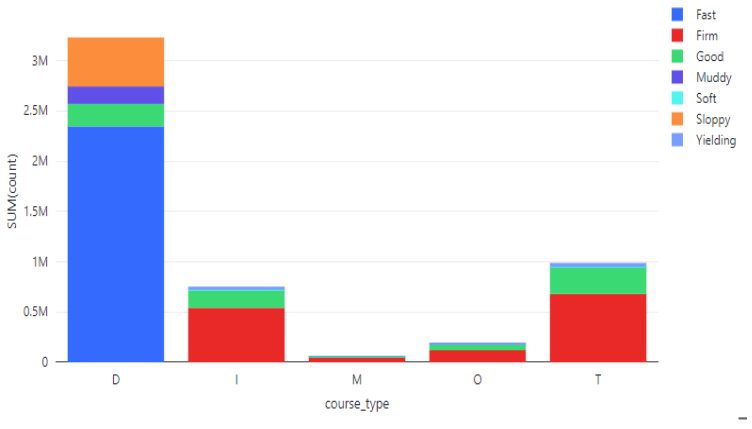

Figure 2. Course type and Condition of tracks during 2019 New York Derby

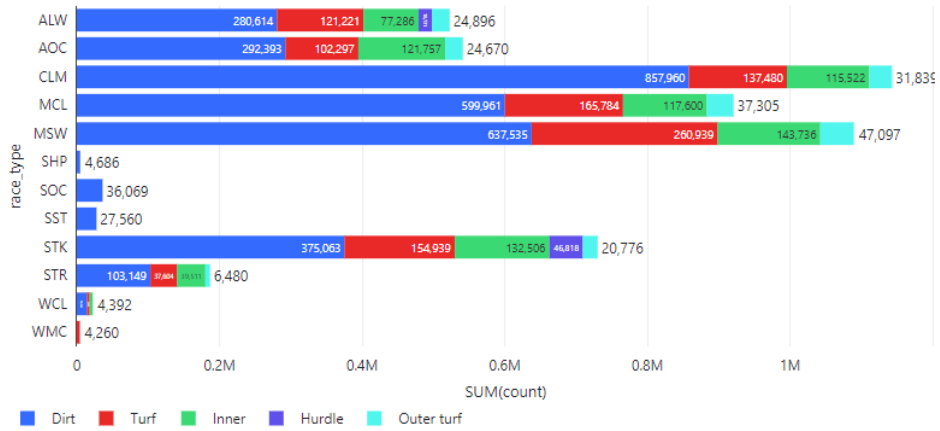

Figure 3. Type of races that took place in New York during 2019.

The distance of the races ranged from 4.5 to 20 furlongs, with 6 furlongs being the most common distance run by horses during the races conducted in New York in 2019, and just a few being over 10 furlongs (see Figure 4).

We also observed that there is a very small subset of jockeys that have an unusually high weight_carried value (see Figure 5). The weight of a jockey usually ranges from 108 to 118 lb (49 to 54 kg). Those high weights are caused by jockeys in races called steeplechase races, where horses and jockey jump over fences throughout the race.

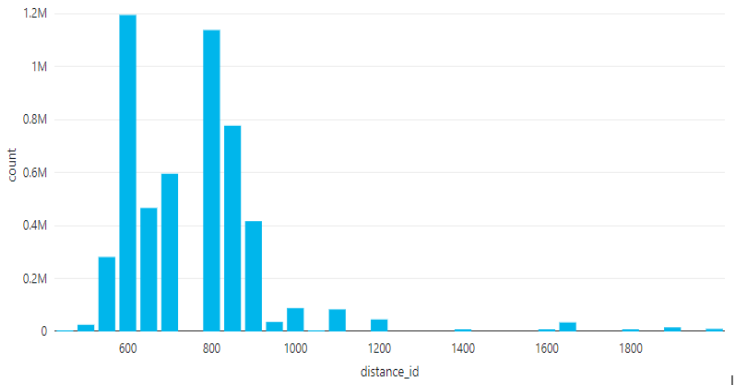

Figure 4. Distance of races run during 2019 in New York Derby

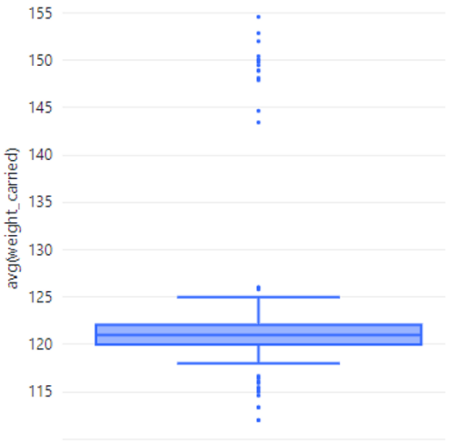

Figure 5. Weight carried by jockey/horse

The top 5 winning jockeys are Manuel Franco, Jose Lezcano, Junior Alvarado, Irad Ortiz Jr, and Jose L. Ortiz. On top of that, Manuel Franco, Jose Lezcano, Junior Alvarado, and Irad Ortiz Jr are also the jockeys that arrived in the top 3 positions the most during all the races they run in 2019 (see Tables 2 and 3).

Manuel Franco won mostly at the Aqueduct race, his performance on Saratoga was not as good as at the other two races. Jose L. Ortiz had a similar performance as Manuel Franco winning most of the time at Aqueduct and having a less performing race in Saratoga (See Appendix 3).

Table 2. Top 5 jockeys that won first place

jockey	position_at_finish	count
Manuel Franco	1	74609
Jose Lezcano	1	72215
Junior Alvarado	1	57375
Irad Ortiz Jr.	1	52930
Jose L. Ortiz	1	47971

Table 3. Top 5 jockeys in the 3 top finish positions

jockey	count
Manuel Franco	218579
Jose Lezcano	171444
Irad Ortiz Jr.	150605
Dylan Davis	149462
Junior Alvarado	141537

Finally, from the dataset *horse_ids,* and *horse_names*, we observed that Rhode Island and Newly Minted are the top two horses that arrived the most in the first 3 positions during the derby races, being Rhode Island the horse that won the most the first place (see Figure 5). Nevertheless, the information about the name of horses is very limited and sometimes owners prefer to keep this information private, thus, there could be other horses with higher counts at a first position not being accounted for in this graph.

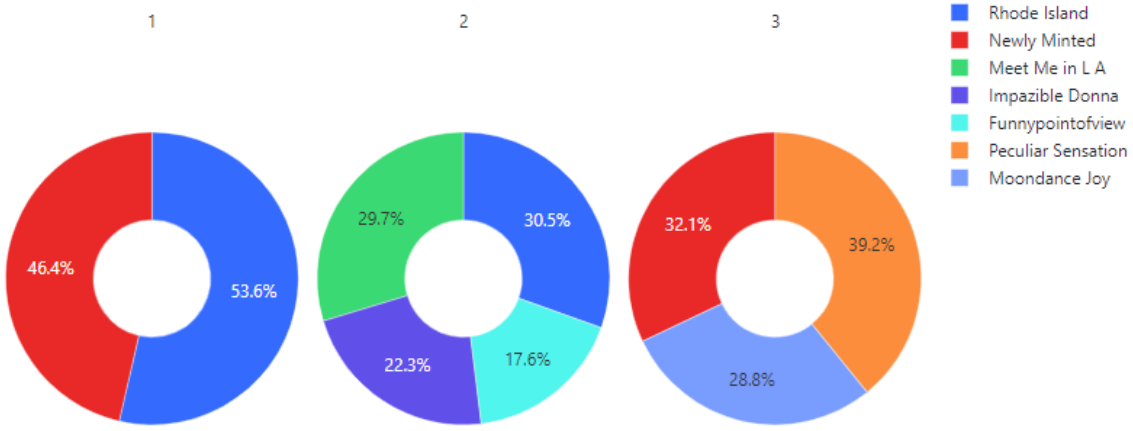

Figure 5. Percentage of times a horse arrived in 1st, 2nd, and 3rd position during New York's Derby in 2019

Machine Learning with Spark MLlib

Predictors

Initially we excluded variables whose information was better represented by other variables. We excluded features race_date, race_number, and program_number because this information was already available in column ID_horse. The post_time variable was excluded because we considered the time of start not to be significant to determine whether a horse/jockey will win a race or not. Finally, variables trakus_index, latitude, and longitude were also excluded from the model because they represent a continuous feed of the position of the horse during the race, and therefore, there are several values for the same race. This information is valuable for running simulations. However, the variables as they are provide limited information to determine the position each horse will finish at the end of the race, and using them as predictors is out of the scope of this project.

Initial predictors for the model: ['ID_horse', 'track_id', 'jockey', 'position_at_finish', 'distance_id', 'course_type', 'track_condition', 'run_up_distance', 'race_type', 'purse', 'weight_carried', 'odds', 'incident_type', 'days_from_injury']

Target Variable

The target variable was predicting the winning horse of the race. Therefore, we conceptualized this idea by creating a new column called target. We transformed the column position_at_finish and set Y= 1 when the horse arrives in the 1st position, and Y = 0 when the horse finishes at any position other than 1. We then dropped the column position_at_finish from the data frame to avoid overfitting the model.

Target variable: [‘target’] - Column derived from position_at_finish

Feature Matrix for ML Algorithms

After defining the features and the target variable, we proceeded to transform our data frame. We started by converting categorical variables track_id, distance_id, jockey, course_type, track_condition, race_type, and incident_type into categorical vectors using StringIndexer and OneHotEncoder functions from the Spark ML library.

We continued by creating a pipeline to index all our features into one vector and dropped the variables that were being duplicated. Then, we prepared our feature matrix using VectorAssembler and split our dataset with a proportion of 0.7/0.3.

We then proceeded to calculate the relation between the most frequent label and the less frequent label to account for any imbalance in the dataset. The ratio between labels gave us a result of 6 which we considered significantly high, thus, we decided to balance the training set using undersampling.

Finally, we used the function StandardScaler to scale our features to avoid the algorithm being biased towards features which have higher values in magnitude. Once our dataset was prepared, we proceeded to train the models and evaluate them.

Random Forest Classifier

We created a pipeline using ParamGridBuilder and CrossValidation to tune and estimate the best parameters to fit our model while using a Random Forest Classifier. Given the size of our training dataset, the command took around 1.65 hours to complete. The results obtained from the CrossValidation pipeline set the feature numTrees to 20. Thus, we defined our final model for the Random Forest classifier as follows:

```
rf = RandomForestClassifier(labelCol="target", featuresCol="ScaledFeatures", numTrees= 20, maxDepth= 5)
```

Once the model was fitted to our training vector, we transformed the test dataset and evaluated its accuracy using RegressionEvaluator to calculate metrics RMSE, MAE, and R-squared, and MulticlassClassificationEvaluator to calculate the accuracy of the model. The results can be observed in Figure 6 below.

```
RMSE:  0.01499906258788147
MAE:  0.0002249718785151856
R-squared:  0.9979723565368124
```

▸ (1) Spark Jobs

```
Accuracy of RandomForestClassifier is = 0.9997750281214848
```

Figure 6. Evaluation metrics and Accuracy of model Random Forest Classifier

From the results obtained with the Random Forest Classifier we observed a very high accuracy of over 99%. Moreover, the metrics calculated also show a good fitting of the model to the data. By RMSE and MAE being close to zero, it is observed that the data points are close to the model's predicted values generating an accurate response of the model to predict whether a horse/jockey will win a race or not. Nevertheless, we could be obtaining a model that overfits the training data.

Naive Bayes

We created once again a pipeline using ParamGridBuilder and CrossValidation to tune and estimate the best parameters to fit our model while using a Naive Bayes model. In this case, the results obtained from the CrossValidation pipeline set the feature modelType=multinomial, which is the default value of this parameter. Thus, we defined our final model for the Naive Bayes classifier as follows

```
nb = NaiveBayes(labelCol="target", featuresCol="ScaledFeatures")
```

The model was then evaluated using the RegressionEvaluator and Multiclass Classification Evaluator and yielded the results observed in Figure 7.

▸ (1) Spark Jobs

```
Accuracy of Naives Bayer is = 0.5686012897487214
```

Figure 7. Evaluation metrics and Accuracy of model Naive Bayes

From the results obtained with the Naive Bayes model we observed a low accuracy of around 50%. The low accuracy obtained with this model could be due to high number of categorical variables present in the Derby dataset. It is known that Naive Bayes model can be not reliable if there are significant differences in the attribute distributions, which is the case with large number of categorical variables.

Conclusion

This report showed that by training a Random Forest Classifier model on the Big Data Derby dataset, we were able to predict whether a horse will win a race or not, with high accuracy of over 90%, RMSE, MAE metrics close to zero, and an R-squared value over 0.99. The Naive Bayes model on the other hand showed a significantly lower performance at predicting the target variable.

On the other hand, Spark provides an efficient and convenient environment for end to end machine learning applications. The computation time of different activities involved in machine learning pipelines such as data cleaning or model training is significantly lower than using other tools such as pandas or Scikit learn.

Overfitting

We observed that RandoForestClassifier might be overfitting the data and could not be as accurate as presented in this report when evaluating new datasets. The overfitting could be due to a large amount of irrelevant information in the dataset; noisy data that might have arrived from adding two extra datasets to the data frame or not having performed a feature selection stage where only features relevant to predicting the variable target were selected. This might have caused the models to learn the noise within the training data and become extremely accurate at predicting the target variable in this specific dataset.

To evaluate whether the model is overfitting the dataset or not, a new evaluation sample could be tested or a K-fold CrossValidation method could be applied where dividing the training set into K equally sized subsets or folds and evaluating the models multiple times. In addition, other techniques for balancing the data could be tested as well as regularization techniques.

Limitations and Future Development

The Big Data Derby dataset contained a big amount of categorical variables which were hard to handle with pyspark. Moreover, the platform Databricks community limits users in the number of drivers connected to the cluster making the processing of data slow.

For future development of this project, handling categorical variables for feature selection could be implemented to better define the relevant features in predicting the label without overfitting the training dataset. We attempted using the Chi-Square Selector method from pyspark library setting up the parameters to present predictors with a 95% probability (fpr=0.05) of inferring the position_at_finish value (original column from which the target variable is derived). From this method, we obtained the following ranking:

Table 4. Feature Selection Ranking

Rank	Variable
1	Odds
2	Jockey
3	course Type
4	Track Condition
5	Race Type
6	Incident Type
7	Weight Carried
8	Track ID
9	Purse
10	distance ID
11	Run Up Distance
12	Days from Injury

Nevertheless, we were not able to conduct the same approach after encoding the categorical variables. Thus, for future development of this project, we would suggest using a well-fitted feature selection method to reduce the noise data in the training dataset and avoid an overfitted model.

Another limitation found while working with MLib on spark was the number of machine learning algorithms available. For future work, we suggest exploring other libraries and testing classifiers which include parameters for regularization so that we have more options for tackling overfitting.

Appendix 1. Variables in each file for the Big Data Derby dataset in the original Kaggle competition

File	Variable	Description
nyra_start_table.csv	track_id	3 character id for the track the race took place at. AQU: Aqueduct, BEL: Belmont, SAR: Saratoga.
	race_date	date the race took place. YYYY-MM-DD.
	race_number	Number of the race. Passed as 3 characters but can be cast or converted to int for this data set.
	program_number	Program number of the horse in the race passed as 3 characters. Should remain 3 characters as it isn't limited to just numbers. Is essentially the unique identifier of the horse in the race.
	weight_carried	An integer of the weight carried by the horse in the race.
	jockey	Name of the jockey on the horse in the race. 50 character max.
	odds	Odds to win the race passed as an integer. Divide by 100 to derive the odds to 1.
	position_at_finish	An integer of the horse's finishing position
nyra_race_table.csv	track_id	3 character id for the track the race took place at. AQU: Aqueduct, BEL: Belmont, SAR: Saratoga.
	race_date	date the race took place. YYYY-MM-DD.
	race_number	Number of the race. Passed as 3 characters but can be cast or converted to int for this data set.
	distance_id	Distance of the race in furlongs passed as an integer. Example - 600 would be 6 furlongs.
	course_type	The course the race was run over passed as one character. M - Hurdle, D - Dirt, O - Outer turf, I - Inner turf, T - turf
	track_condition	The condition of the course the race was run on passed as three characters. YL - Yielding, FM - Firm, SY - Sloppy, GD - Good, FT - Fast, MY - Muddy, SF - Soft.
	run_up_distance	Distance in feet of the gate to the start of the race passed as an integer.
	race_type	The classification of the race passed as as five characters. STK - Stakes, WCL - Waiver Claiming, WMC - Waiver Maiden Claiming, SST - Starter Stakes, SHP - Starter Handicap, CLM - Claiming, STR - Starter Allowance, AOC - Allowance Optionl Claimer, SOC - Starter Optional Claimer, MCL - Maiden Claiming, ALW - Allowance, MSW - Maiden Special Weight.
	purse	Purse in US dollars of the race passed as an money with two decimal places.
	post_time	Time of day the race began passed as 5 character.
nyra_tracking_table.csv	track_id	3 character id for the track the race took place at. AQU: Aqueduct, BEL: Belmont, SAR: Saratoga.
	race_date	date the race took place. YYYY-MM-DD.
	race_number	Number of the race. Passed as 3 characters but can be cast or converted to int for this data set.
	program_number	Program number of the horse in the race passed as 3 characters. Should remain 3 characters as it isn't limited to just numbers. Is essentially the unique identifier of the horse in the race.
	trakus_index	The common collection of point of the lat / long of the horse in the race passed as an integer. From what we can tell, it's collected every 0.25 seconds.
	latitude	The latitude of the horse in the race passed as a float.
	longitude	The longitude of the horse in the race passed as a float.

Appendix 2. Variables in added files for the Big Data Derby Dataset

File	Variable	Description
horse_ids.cvs	track_id	Track the race took place Aqueduct, Belmont, Saratoga.
	race_date	date the race took place. YYYY-MM-DD.
	race	Number of the race. Passed as 3 characters but can be cast or converted to int for this data set.
	program_number	Program number of the horse in the race passed as 3 characters. Should remain 3 characters as it isn't limited to just numbers. Is essentially the unique identifier of the horse in the race.
	horse_id	ID of each horse
	finishing_place	The position the horse finished in.
horse_names.cvs	horse_id	ID of each horse
	horse_name	Name of each horse
equine.csv	year	Year of the incident
	incident_date	Date the the incident took place.
	incident_type	Type of the incident
	track	Track in which the incident happened
	inv_location	Track in which the incident happened
	racing_type_description	Description of the type of race where the incident happened
	division	Division where the incident happened
	weather_conditions	Weather conditions when the incident happened
	horse	Horse invovled in the incident
	trainer	Trainer of the horse
	jockey_driver	Jockey driving the horse when the incident happened
	incident_description	Description of the type of incident
	death_or_injury	Result from the injury on the animal

Appendix 3. Count of jockeys that won first place by track during Derby races in New York 2019