**Artificial Intelligence Capstone Project1**

Student name: 吳權祐 Student ID: 110550014

1. **Introduction**

The 2024 Paris Olympics are coming just around the corner, as one of the most renowned sporting events globally, the result of the competition is focused by the entire world. I want to see if we can predict the result of the game. Based on this motivation, I collected the historical records of men’s 100 meters as my dataset and explored how machine learning and artificial intelligence can use the features to make predictions and improve sports.

Github repo with complete code:

<https://github.com/ailuropodaWu/NYCU-AI-Capstone/tree/main/Project1>

1. **Dataset**
   1. **Introduction**

This is a dataset on the men’s 100 meters events in the Olympics from 1948 to 2020 (without 1952 since there is no wind information from that year).

The attributes in this dataset correspond to the information about athletes and other relevant details about the events. The label of the dataset represents the performance of each athlete in a competition.

* + 1. **Compositions**

Each row represents a record with following attributes about a competitor in a single game.

**Attributes List**

***Name***: string , name of the competitor, unuseful feature

***Nation***: int , code of the nation where the competitor from according to the dictionary (sorted by nation frequency in the dataset) (Appendix 1.2).

***Weight*** *(kg)*: float , weight of the competitor

***Height*** *(m)*: float , height of the competitor

***BMI***: float , calculated by the formula

***Age***: float , age of the competitor, calculated from the birthday to the first day the event start

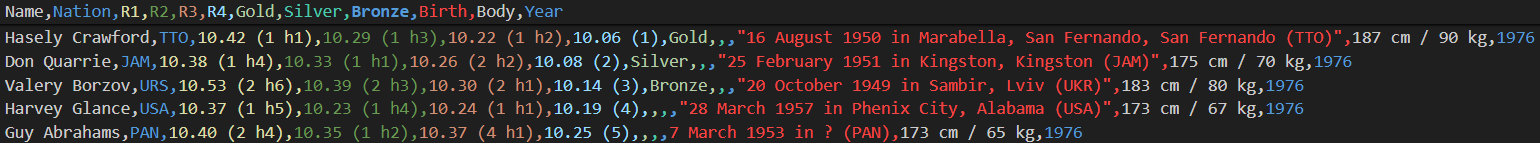
***Year***: int , the year of the event

***Round***: int , round in the competition, range from 1 – 4 (4 means final)

***Wind*** *(m/s)*: float , wind information of the game

***isHometown***: bool , whether the venue of the event is the hometown of the competitor or not

***Label*** *(s)*: float , performance of the competitor in the game

* + 1. **Data examples**
* ****Raw data
* **一張含有 文字, 螢幕擷取畫面 的圖片

  自動產生的描述**Corresponding data
  + 1. **Amountand conditions**

There are totally **2090** data in the dataset, and the label (performance) of data is conditioned to be under **11.5** to avoid outlier data. And the dataset consists of **921** athletes and **180** nations.

* 1. **Collection**

The stages of collection consist of data scraping and data processing. And all these processes are performed on my laptop using the Visual Studio Code IDE.

* + 1. **Scrape**

All data is scrapped from the internet, particularly from the ***Olympedia*** website. I wrote a ***sraper.py*** (Appendix 1.1) scripts using the *requests* and *BeautifulSoup* packages to extract the raw data of all competitors in each competition and the wind information of the competition separately.

* + 1. **Process**

Then, I used a ***generate\_train\_data.ipynb*** (Appendix 1.2) to process the raw data and generate the processed data. The packages used in the files are *pandas, re, datetime* and *calendar*.

In this stage, it was necessary for me to preprocess the data to make the following steps easier. First, I transformed any data with null values to match the format of the other values in the same columns. Then, some of the features were extracted from the raw data, such as *Weight* and *Height* from the *Body* column. Lastly, I dropped some useless attributes, such as *Gold*, *Silver*, and *Bronze*, which are used to record who got the medal. I also processed some information in the data. For example, the original *Birth* data contained information about the birthplace, and I processed it to preserve only the pure birthday.

The next step is to process the data into a specific format. Since the format of the raw data contains all results for one athlete in an event in a specific year, I separately extracted and filtered the results of the games the athlete attended that year.

In the final step of this stage, I created three new attributes extended from the original columns. *Age* was calculated based on the athlete’s birthday and the start date of the event. *isHometown* was a Boolean value that determined by whether the *Nation* of the athlete matched the country of the venue. And *BMI* attribute was calculated using the formula mentioned previously. After completing these steps, I retained the desired columns to obtain my own dataset.

* 1. **External source**
     1. **Packages**
* [requests](https://requests.readthedocs.io/en/latest/) (https://requests.readthedocs.io/en/latest/#)
* [BeautifulSoup](https://requests.readthedocs.io/en/latest/) (https://requests.readthedocs.io/en/latest/#)
* [pandas](https://pandas.pydata.org/docs/index.html) (https://pandas.pydata.org/docs/index.html)
* [re](https://docs.python.org/3/library/re.html) (https://docs.python.org/3/library/re.html)
* [datetime](https://docs.python.org/3/library/datetime.html) (https://docs.python.org/3/library/datetime.html)
* [calendar](https://docs.python.org/dev/library/calendar.html) (https://docs.python.org/dev/library/calendar.html)
  + 1. **Websites**
* [Olympedia men’s 100 meters events page](https://www.olympedia.org/event_names/40) (https://www.olympedia.org/event\_names/40)

1. **Methods**

Since the label in this dataset consists of continuous floating numbers, it appears to be a regression problem for prediction. I performed three types of supervised linear regression algorithms and two unsupervised method. The three supervised method are respectively closed-form linear regression, support vector machine (SVM) and deep-learning based model, while the unsupervised methods are both based on the nearest neighbor search problem. All algorithms are implemented in the ***algorithm.ipynb*** (Appendix 1.3) using the ***scikit-learn*** library.

* 1. **Supervised learning**
     1. **Closed-form linear regression**

The algorithm uses Ordinary Least Squares (OLS) method to fit the line to the data, finding all coefficients and the intercept by minimizing errors.

* + 1. **Support Vector Machine**

The core concept of Support Vector Machine (SVM) regressor is to find the best kernel function to transform data to a hyperplane that separates data points. I employed the SVM regressor with a polynomial kernel of degree 20 as one of my methods.

* + 1. **Deep-learning based**

一張含有 寫生, 對稱, 樣式 的圖片

自動產生的描述I used a Multi-layer Perceptron (MLP) as the regressor in my project. The network architecture includes 4 hidden layers, with each layer comprising 10 nodes, as illustrated below.

* 1. **Unsupervised learning**

The core concept of regression is to predict continuous value from the data. In unsupervised learning, I implemented algorithms based on the Nearest Neighbor Search (NNS) problem to find the nearest neighbor of the data and predict its label with the value from the neighbor. Specifically, I used KD-Tree algorithm for this purpose.

1. **Experiments**
   1. **Experiments detail**
      1. **Data split**

In the real-world, people focus on using historical data to predict the results of upcoming Olympic games in a specific year. To simulate this scenario, I divided the original dataset based on the *Year* attributes. The training data includes information from 1948 to 2008, while the testing data with **151** rows comprises data from 2016 and 2020(2021).

* + 1. **Experiment setting**

In the project, I conducted several experiments using different training datasets to explore the effects of data quantity and quality, and compare the performance of the algorithms. Here are the four types of training datasets.

* ***Complete***: the whole dataset

*(1939 data)*

* ***Filtered***: each athlete from a specific year is recorded

*(1088 data)* only once with their best performance in that

year

* ***Thousand***: call ***train\_test\_split*** from ***scikit-learn*** with

*(999 data)*  *train\_size* = 1000 / (numbers of rows in

complete training data)

* ***Hundred***: call ***train\_test\_split*** from ***scikit-learn*** with

*(100 data) train\_size* = 100 / (numbers of rows in complete

training data)

* + 1. **Evaluation metrics**

Here are three types of metrics I used in the experiments to evaluate the performance.

* ***score***: a score calculated by the below formula, it is used to

get the coefficient of determination of the

一張含有 文字, 字型, 筆跡, 白色 的圖片

自動產生的描述prediction.

* ***RMSE***: calculated by the below formula, it represents the

*一張含有 字型, 文字, 印刷術, 白色 的圖片

自動產生的描述(RMSD)* average error of the prediction.

* ***Accuracy***: the accuracy rate under a specific error.
  1. **Experiments result**

The underlined numbers are the best performances from the same training data, while the bold ones are from the same algorithm.

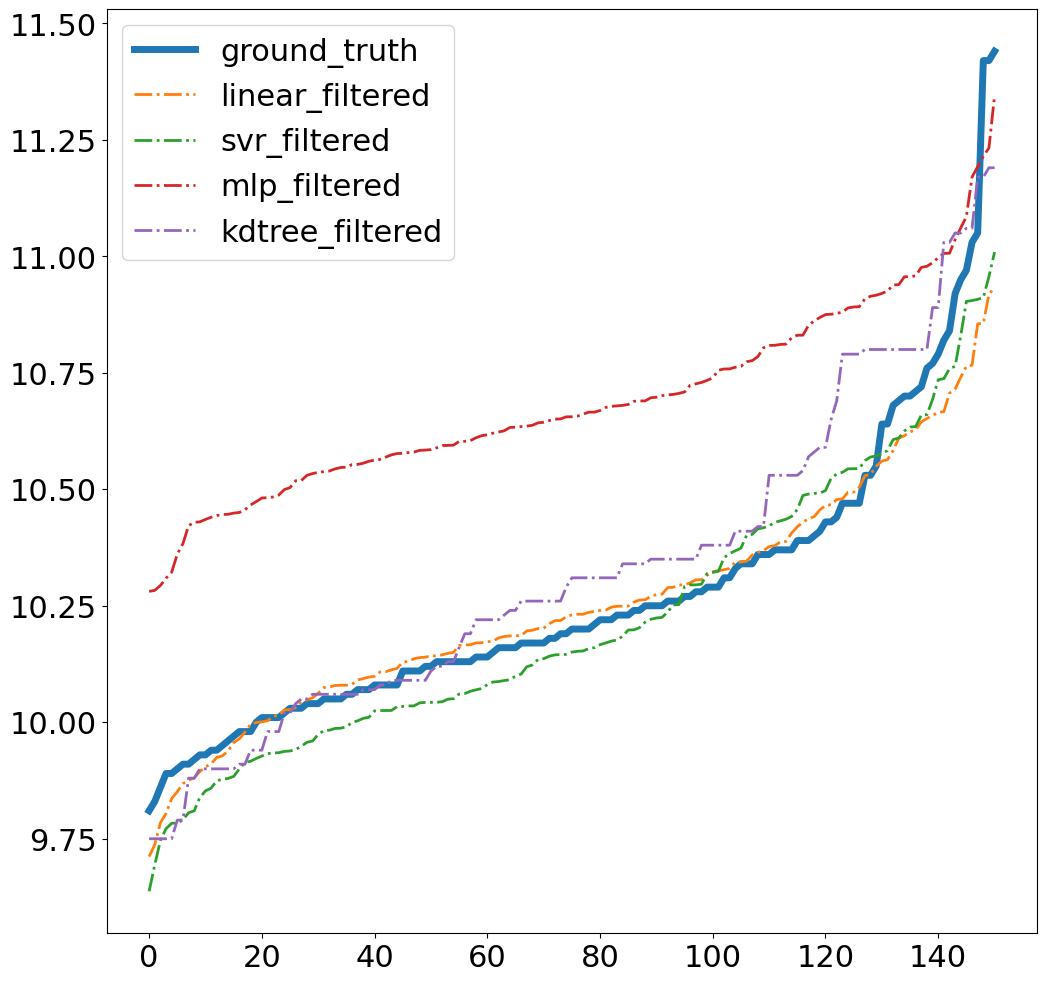
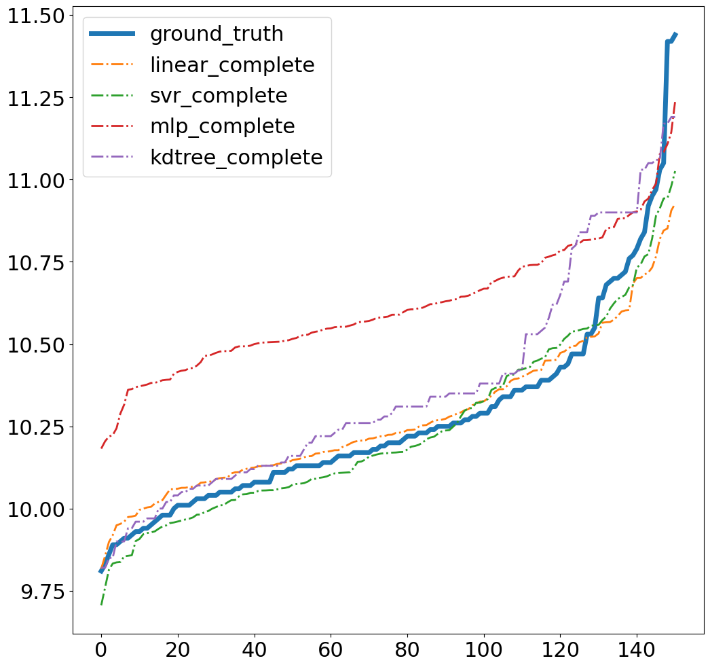
It appears that the closed-form linear regression algorithm consistently performs best in predicting results across different datasets related to the field, whereas deep learning-based methods show poor performance. Furthermore, regarding datasets, data size significantly affects training, while whether the data are filtered or not seems not as critical.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***algorithms*** | ***Linear*** | ***SVR*** | ***MLP*** | ***KD-Tree*** |
| ***Complete*** | | | | |
| ***score*** | 0.563 | **0.531** | -1.306 | - |
| ***RMSE*** | 0.202 | **0.210** | 0.465 | 0.370 |
| ***ACC (0.1)*** | 0.430 | **0.384** | 0.060 | **0.265** |
| ***ACC (0.01)*** | **0.073** | 0.033 | 0.007 | **0.046** |
| ***Filtered*** | | | | |
| ***score*** | **0.606** | 0.507 | -1.883 | - |
| ***RMSE*** | **0.192** | 0.215 | 0.520 | **0.367** |
| ***ACC (0.1)*** | **0.464** | 0.344 | 0.073 | 0.245 |
| ***ACC (0.01)*** | 0.046 | 0.013 | 0.007 | 0.040 |
| ***Thousand*** | | | | |
| ***score*** | 0.575 | 0.518 | **-1.243** | - |
| ***RMSE*** | 0.200 | 0.212 | **0.459** | 0.380 |
| ***ACC (0.1)*** | 0.450 | 0.338 | **0.079** | 0.232 |
| ***ACC (0.01)*** | 0.066 | **0.053** | 0.007 | 0.020 |
| ***Hundred*** | | | | |
| ***score*** | 0.525 | 0.269 | -3.278 | - |
| ***RMSE*** | 0.202 | 0.262 | 0.634 | 0.445 |
| ***ACC (0.1)*** | 0.331 | 0.305 | 0.066 | 0.126 |
| ***ACC (0.01)*** | 0.026 | 0.046 | **0.013** | 0. |

* 1. **Visualized results**

I visualized the experiments result using the ***matplotlib*** library for additional analysis. Complete visualized results could be found in Appendix 2.

Closed-form linear regression and SVM regressor are stable algorithms that closely approximate the ground truth.



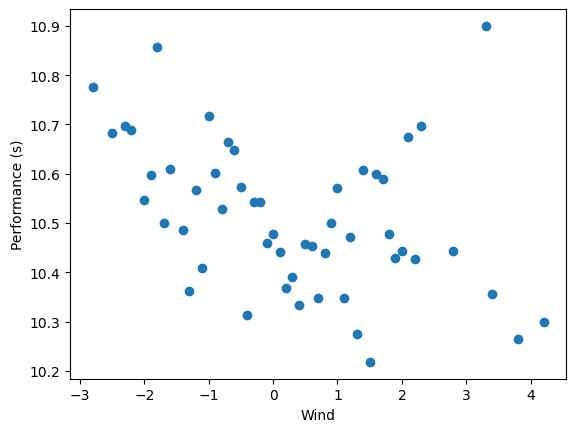
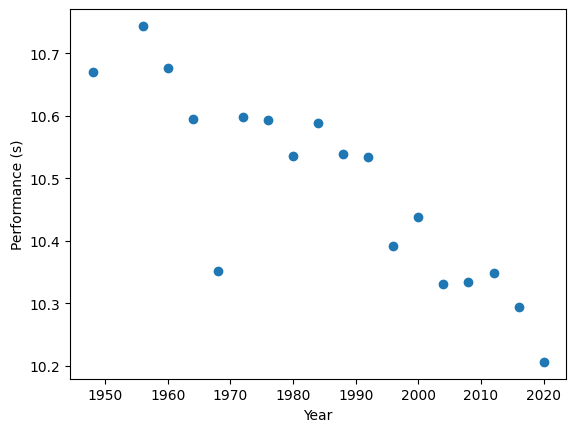
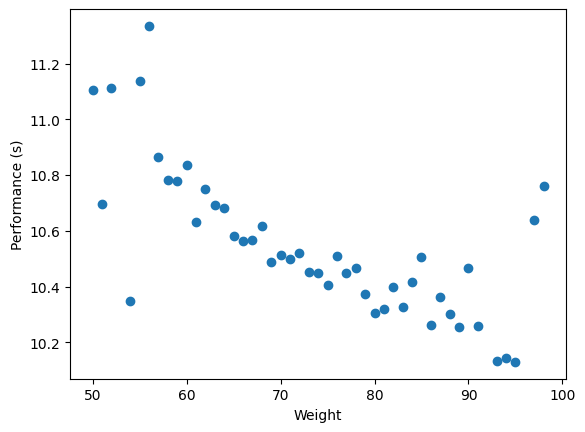
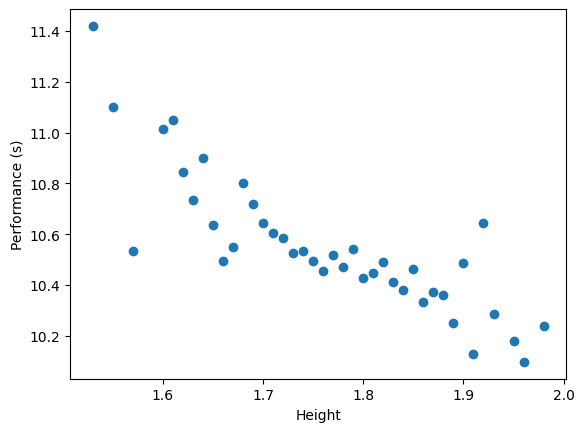
一張含有 文字, 圖表, 螢幕擷取畫面, 行 的圖片

自動產生的描述The predictions from unsupervised learning become more discrete when the amount of data decreases. This is due to the reduced number of neighbors in the training data available for prediction.

1. **Discussion**
   1. **Conclusion**

Before the experiments, I expected deep-learning based methods to achieve the best performance, however, they actually showed the poorest performance. This led to the understanding that the issue might be related to the size of the data. With only about 2,000 data available, it seems insufficient for training a deep-learning model effectively. Something I didn’t expect is the outperformance of closed-form linear regression. I believe this suggests that sports, especially events like men’s 100 meters, are highly scientific and analyzable.

I have performed some feature engineering and discovered that most features in the dataset are highly correlated with the label. Specifically, features like *Height*, *Weight*, *Year*, and *Wind* exhibit high negative correlation with the average of performances, as shown in the figures below.



* 1. **Future experiments**

If there is more time available, I would like to conduct experiments related to hyperparameters and dimension reduction. I am curious about how hyperparameters impact the performance of deep-learning methods and what architecture would be suitable for models learning from limited data. Additionally, I am interested in exploring how dimension reduction affects performance and determining the optimal amount of feature reduction required.

* 1. **Gained knowledge**

Sports is an interesting and scientific field that is suitable for research. However, as seen in the results, the performance of the algorithms is still not optimal. In the future, we could explore other methods for data collection to create a stronger dataset or conduct further research on feature engineering to better understand the relationship between the data and the label.

1. **References**

[https://requests.readthedocs.io/en/latest/#](https://requests.readthedocs.io/en/latest/)

[https://requests.readthedocs.io/en/latest/#](https://requests.readthedocs.io/en/latest/)

[https://requests.readthedocs.io/en/latest/#](https://requests.readthedocs.io/en/latest/)

<https://pandas.pydata.org/docs/index.html>

<https://docs.python.org/3/library/re.html>

<https://docs.python.org/3/library/datetime.html>

<https://docs.python.org/dev/library/calendar.html>

<https://www.olympedia.org/event_names/40>

<https://scikit-learn.org/stable/index.html>

<https://matplotlib.org/>

<https://zhuanlan.zhihu.com/p/67706712>

<https://en.wikipedia.org/wiki/Root-mean-square_deviation>

**Appendix**

1. **Source Code**
   1. **Scrape**

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

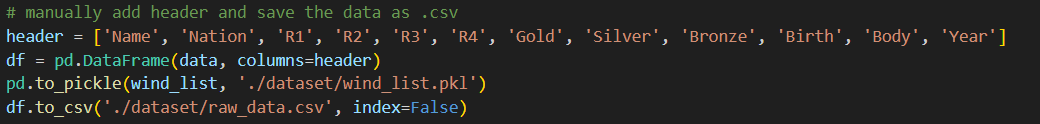
自動產生的描述**

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

****

* 1. **Process**

**一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述**

**一張含有 文字, 螢幕擷取畫面, 陳列 的圖片

自動產生的描述**

**一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述**

**一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述**

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

**Nation Code**

'USA': 0, 'GBR': 1, 'JAM': 2, 'FRA': 3, 'CAN': 4, 'NGR': 5, 'TTO': 6, 'BRA': 7, 'URS': 8, 'POL': 9, 'JPN': 10, 'GHA': 11, 'CUB': 12, 'AUS': 13, 'BAH': 14, 'ITA': 15, 'GDR': 16, 'GER': 17, 'CIV': 18, 'FRG': 19, 'CHN': 20, 'SKN': 21, 'BAR': 22, 'HUN': 23, 'SEN': 24, 'ESP': 25, 'INA': 26, 'VEN': 27, 'KEN': 28, 'GRE': 29, 'RSA': 30, 'BEL': 31, 'POR': 32, 'CMR': 33, 'NAM': 34, 'BUL': 35, 'PAN': 36, 'UGA': 37, 'PAK': 38, 'QAT': 39, 'CGO': 40, 'SUR': 41, 'ANT': 42, 'TPE': 43, 'GAM': 44, 'THA': 45, 'ISV': 46, 'CYP': 47, 'DOM': 48, 'SUI': 49, 'AHO': 50, 'SGP': 51, 'UKR': 52, 'MDV': 53, 'BER': 54, 'IRI': 55, 'MAS': 56, 'PUR': 57, 'BUR': 58, 'NOR': 59, 'MAD': 60, 'LBR': 61, 'SLE': 62, 'KOR': 63, 'KSA': 64, 'MLI': 65, 'TGA': 66, 'ZAM': 67, 'OMA': 68, 'RUS': 69, 'NZL': 70, 'BAN': 71, 'CHI': 72, 'GAB': 73, 'NED': 74, 'BEN': 75, 'ARG': 76, 'AUT': 77, 'ISL': 78, 'CAY': 79, 'MRI': 80, 'SWE': 81, 'MEX': 82, 'FIJ': 83, 'GUI': 84, 'TCH': 85, 'KAZ': 86, 'LES': 87, 'PHI': 88, 'EUN': 89, 'COL': 90, 'GBS': 91, 'GUY': 92, 'HAI': 93, 'HKG': 94, 'TUR': 95, 'URU': 96, 'BOL': 97, 'PLW': 98, 'MLT': 99, 'SEY': 100, 'ESA': 101, 'BRN': 102, 'FIN': 103, 'FSM': 104, 'AZE': 105, 'ZIM': 106, 'TAN': 107, 'SLO': 108, 'KUW': 109, 'MTN': 110, 'PNG': 111, 'ANG': 112, 'ASA': 113, 'ISR': 114, 'VAN': 115, 'IND': 116, 'CAF': 117, 'COK': 118, 'LAO': 119, 'COM': 120, 'SRI': 121, 'SOL': 122, 'SMR': 123, 'TOG': 124, 'IVB': 125, 'SWZ': 126, 'ROU': 127, 'STP': 128, 'VIE': 129, 'WIF': 130, 'SUD': 131, 'AFG': 132, 'HON': 133, 'BOT': 134, 'GEQ': 135, 'GUA': 136, 'PER': 137, 'IRL': 138, 'IRQ': 139, 'LBN': 140, 'ETH': 141, 'BRU': 142, 'LIE': 143, 'BIZ': 144, 'MKD': 145, 'NCA': 146, 'MON': 147, 'MAW': 148, 'CAM': 149, 'ARU': 150, 'CRC': 151, 'CRO': 152, 'CZE': 153, 'UZB': 154, 'VIN': 155, 'TUV': 156, 'ECU': 157, 'ALG': 158, 'EGY': 159, 'ALB': 160, 'EST': 161, 'PLE': 162, 'GEO': 163, 'LUX': 164, 'PAR': 165, 'NEP': 166, 'MOZ': 167, 'MGL': 168, 'MAR': 169, 'MAL': 170, 'LTU': 171, 'GRN': 172, 'LCA': 173, 'LAT': 174, 'KIR': 175, 'JOR': 176, 'SVK': 177, 'GUM': 178, 'KGZ': 179

* 1. **Algorithm**

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

**一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述**

1. **Result Visualization**
   1. **Source code**

**一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述**

* 1. **Complete results**
     1. **Algorithms**

**一張含有 文字, 螢幕擷取畫面, 圖表, 繪圖 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 行, 圖表 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 繪圖, 圖表 的圖片

自動產生的描述一張含有 文字, 圖表, 螢幕擷取畫面, 行 的圖片

自動產生的描述**

* + 1. **Training data**

**一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

自動產生的描述一張含有 文字, 圖表, 行, 螢幕擷取畫面 的圖片

自動產生的描述**