Week 3 – Intro to Summary Statistics

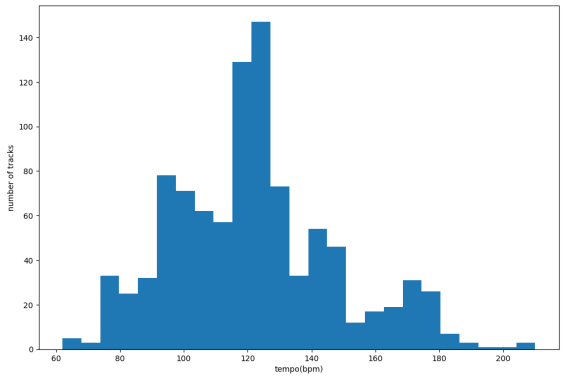
This week, we'll summarise the dataset and some of the core concepts it represents.

First, we're going to check out a dataset of 2019-2022 hot jitterbug music dataset. (FROM: https://www.kaggle.com/sveta151/datasets,Author: Sveta151) I'm going to preprocess the dataset after and integrate it into a DF。

- Tag songs from different years in the dataset with their own year of popularity.
- Replace track_name with a non-repeating year_number.
- Remove irrelevant columns.
- Round all the values to nearest integer

I found that the same track_name could be on the hit list in different years, so there were a lot of problems indexing the data behind it, to the point where the zscore calculation wouldn't bring the data into the df, due to the fact that the same song creates different zscores in different years, so I replaced the trackname with a pattern of non-repeating year_serial numbers to avoid this problem.

After briefly processing the data, we can get a histogram of the distribution of tempo for all songs.



Central Tendency

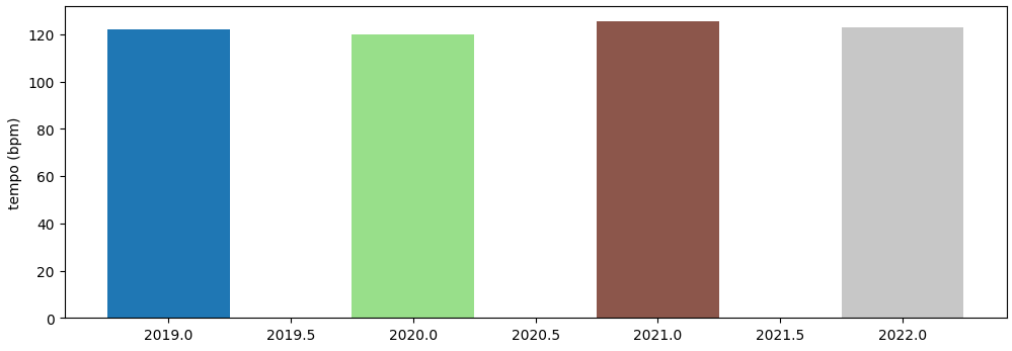
The MEAN, MEDIAN and MODE (most frequent values) of the data were calculated.

They were 122.3, 121.4, 120.0.

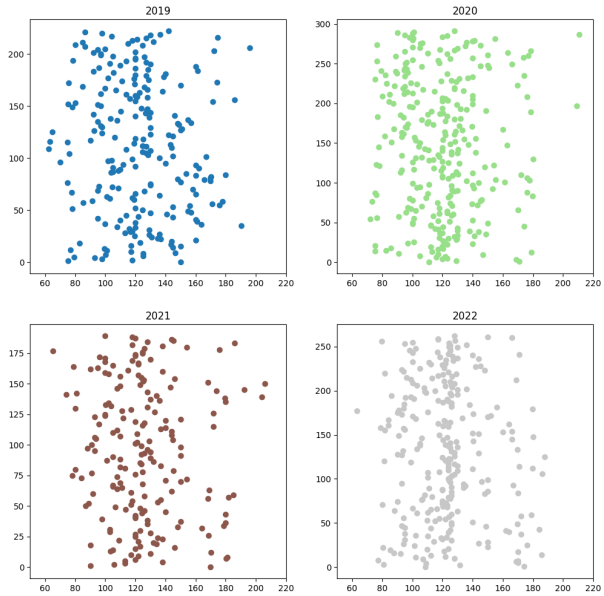
Individual Genres

We can split up these tempo stats into different year and see how they vary between them.

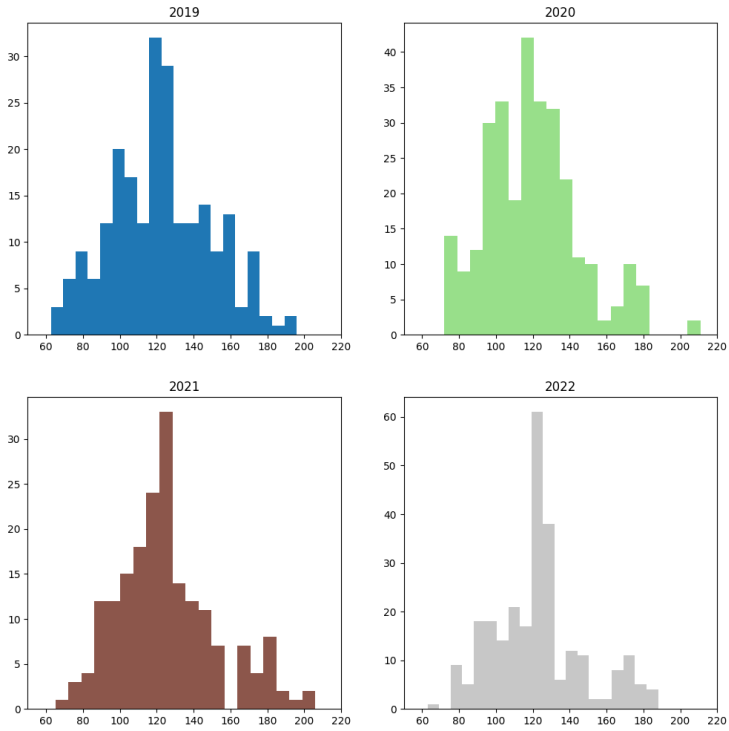
1. We compare the difference in tempo between years by creating new arrays, giving them different colors and calculating the average tempo between years in the same bar chart.



2.Draw the distribution of tempo for different years. It is possible to compare the distribution of tempo in different years.



3. Comparing the tempo distribution of hot music from different years.



4. The median is a more accurate reflection of the dataset than the mean when there are more extremes, so we then attempted to calculate the median for the 2019 song data.

```
#Mean
mean = y2019.mean()
print(mean)
#Median
median = y2019.median()
print(median)
```

✓ 0.0s

```
122.19657847533632
120.045
```

5. Music mode vs. mean and median for 2020.

```
#Mean
mean = y2020.mean()
print(mean)
#Median
median = y2020.median()
print(median)
#Most common 3 tempos
mode = Counter(np.floor(y2020)).most_common(3)[:3]
print(mode)
```

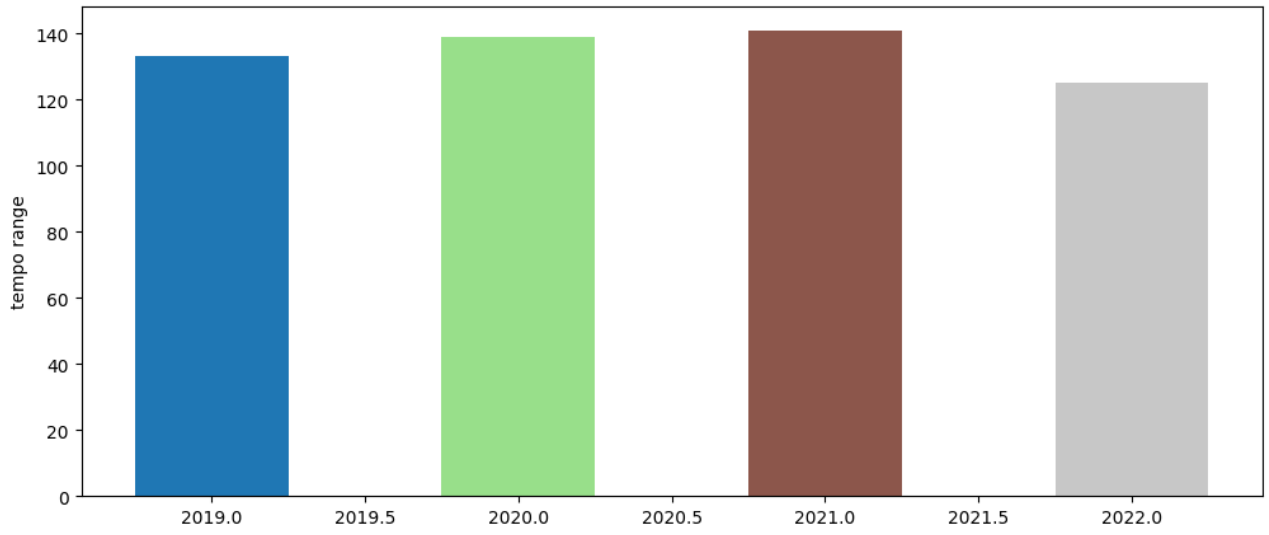
✓ 0.0s Python

```
122.19657847533632
120.045
[(119.0, 12), (129.0, 10), (127.0, 7)]
```

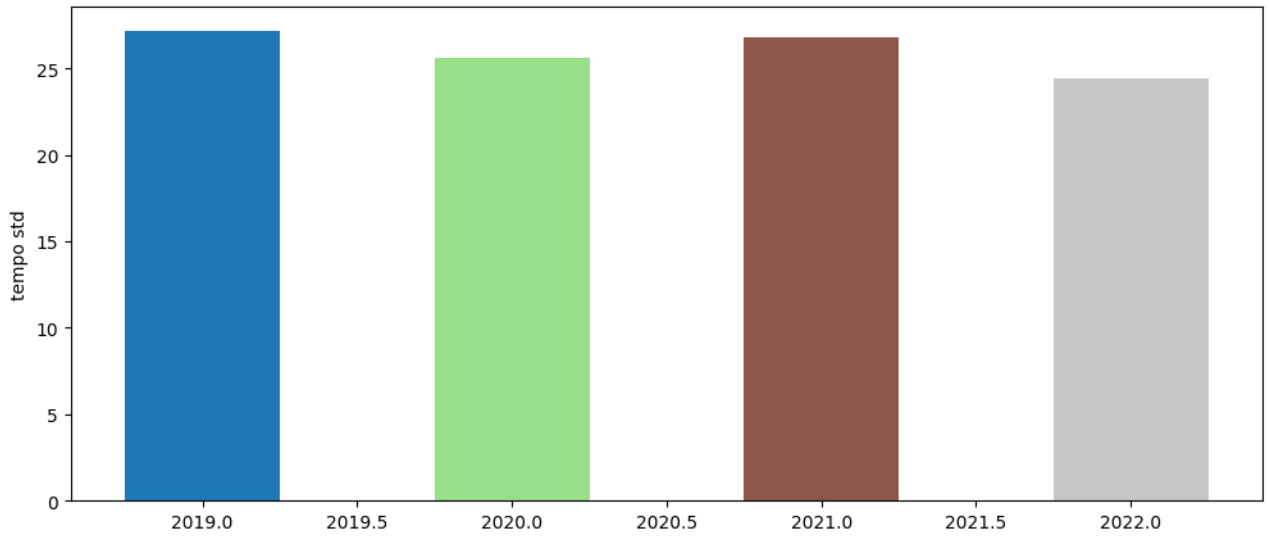
Measures of Variation

We noticed some differences in the distribution of these values across the set. We attribute this to how they vary across the sample. By getting some statistics to describe this variation.

1. Get range from grouped items.

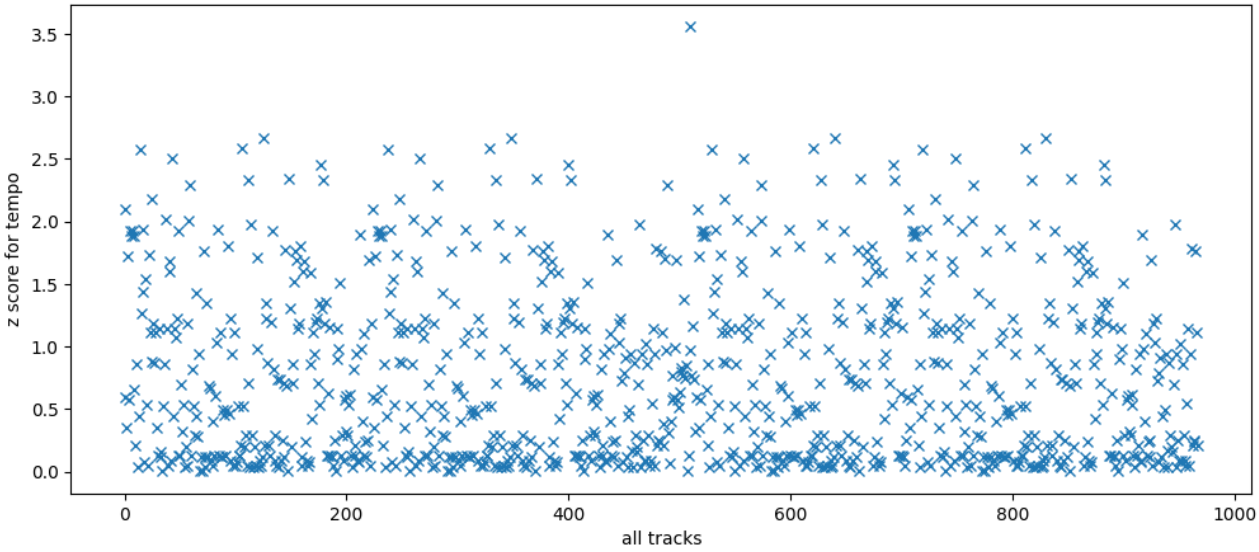


2. Get standard deviation from grouped items.



3. Z Scores

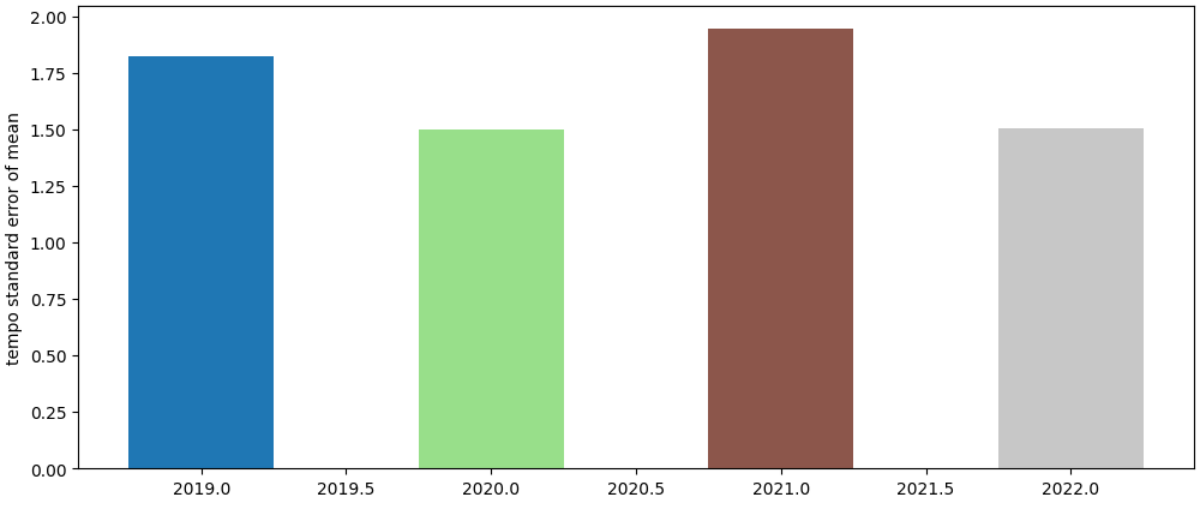
For normally distributed data, 95% of the items should be within 2 standard deviations of the mean. This method of calculating the standard deviation of a measurement from the mean is called a zscore. this can be used in a fairly general way to identify outliers.



Sample Error

There will always be an error between the statistics we report (e.g. the mean) and the true value. We can estimate how confident we can be that the sample is representative of the whole population by using the standard error (std / sqrt (sample size)).

1. tempo standard error of mean



2. Confidence intervals

By calculating confidence intervals, we can see how reliable our estimates of the mean musical tempo are for each year.

```
2019 122.2 +- 5.331
2020 119.8 +- 5.019
2021 125.5 +- 5.259
2022 122.9 +- 4.782
```

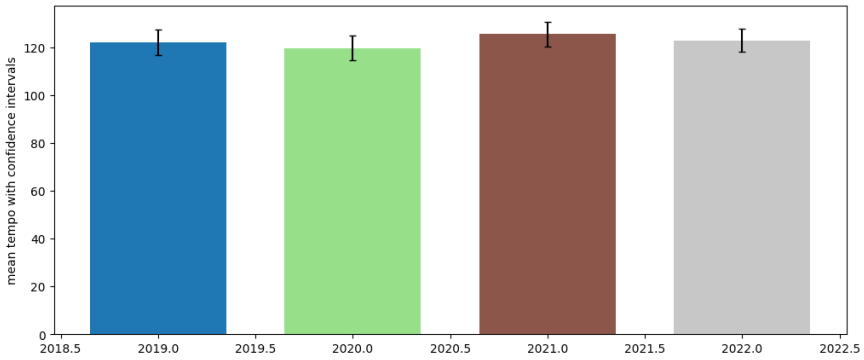
Num_sample:100

```
2019 122.2 +- 3.77
2020 119.8 +- 3.549
2021 125.5 +- 3.719
2022 122.9 +- 3.381
```

Num_sample:200

Adding num_samples reduces the effect of outliers on the confidence interval and improves the accuracy of the estimate.

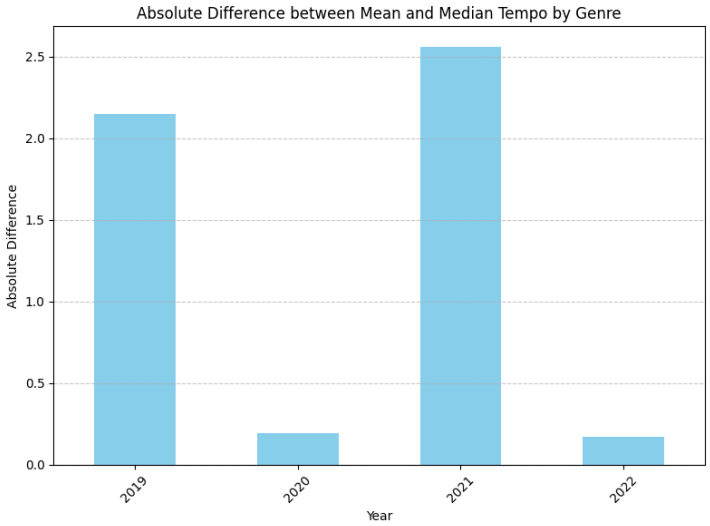
3. Making a Plot with confidence intervals on the mean tempo



Task

1. Define a function to calculate absolute difference

Absolute_difference_tempo=mean_tempo-median_tempo



The absolute difference between the mean and median rhythms is larger in 2019 and 2021, while the absolute difference is smaller in 2020 and 2022

2. Print the filenames of tracks with a z value greater than 3

```
# Print the filenames of tracks with a z value greater than 3
outliers = df[df['zscore'] > 3]['Num']
print(outliers.tolist())
```

✓ 0.0s Python

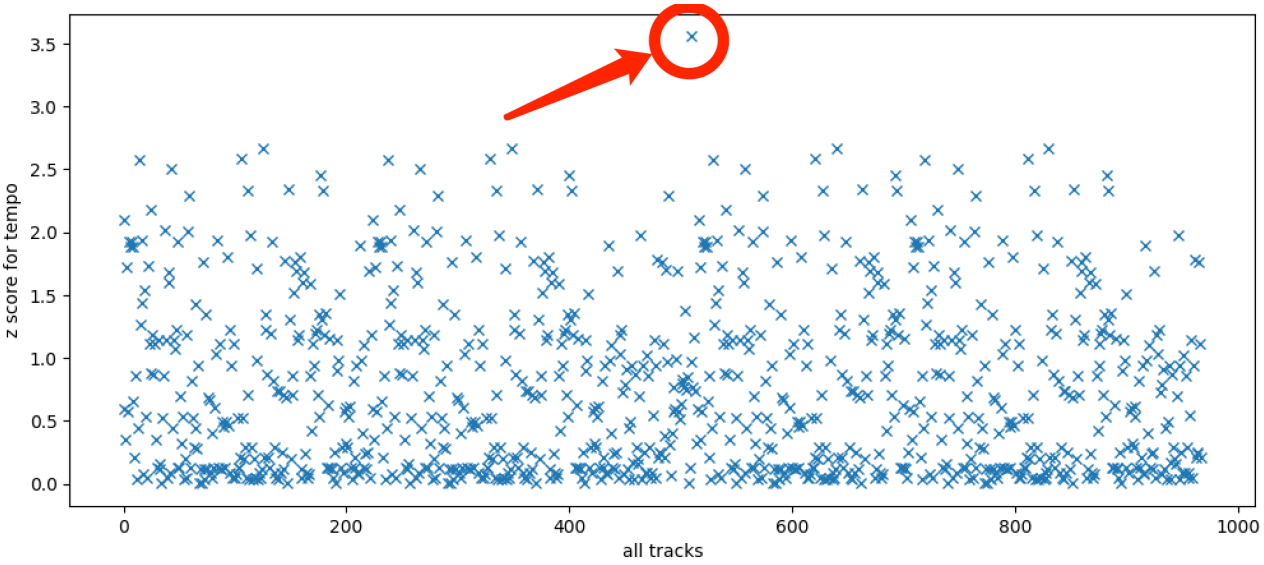
```
['2020.00288']
```

```
# Find "trackname" in the line where the "Num" column is '2020.00288'.
track_name = df_backup.loc[df_backup['Num'] == '2020.00288', 'track_name'].values[0]
print(track_name)
```

✓ 0.0s Python

```
Buttons
```

botton The Z score of this song is greater than 3 representing a larger difference in tempo compared to the entire dataset and may have a particular musical drum beat and rhythm.



We can also find in the graph.

3. Task 3

```
#3.1
num_samplesodd_numbers = [i for i in np.arange(0,100) if i % 2 == 1 ]
num_samplesodd_numbers
#3.2
a = [i * 2 + 1 for i in np.arange(0,50)]
print(a)
#3.3 Sum the list of numbers from 1-i each time
a = [np.sum(np.arange(i)) for i in np.arange(2,12)]
print(a)
#3.4
a = np.random.random(100)
#Use in place conditional
a = [1 if i > 0.5 else 0 for i in a]
print(a)
```

4. Task 4

Change number of samples in Confidence intervals

```
### Confidence intervals
confidence_intervals = []
# num_samples used to be 100
# out put
# 2019 122.2 +- 5.331
# 2020 119.8 +- 5.019
# 2021 125.5 +- 5.259
# 2022 122.9 +- 4.782
num_samples = 200
z = 1.96
for label in Year:
    #Get mean for genre
    mean = mean_tempos.loc[label]
    #Get standard deviation for genre
    std = std_tempos.loc[label]["tempo"]
    #Get confidence range
    dist = z * (std / np.sqrt(num_samples))
    print(label, np.round(mean, 1), "+-", np.round(dist,3))
    confidence_intervals.append([dist,dist])
```

output:

```
2019 122.2 +- 3.77
2020 119.8 +- 3.549
2021 125.5 +- 3.719
2022 122.9 +- 3.381
```

The width of the confidence interval decreases

Summary

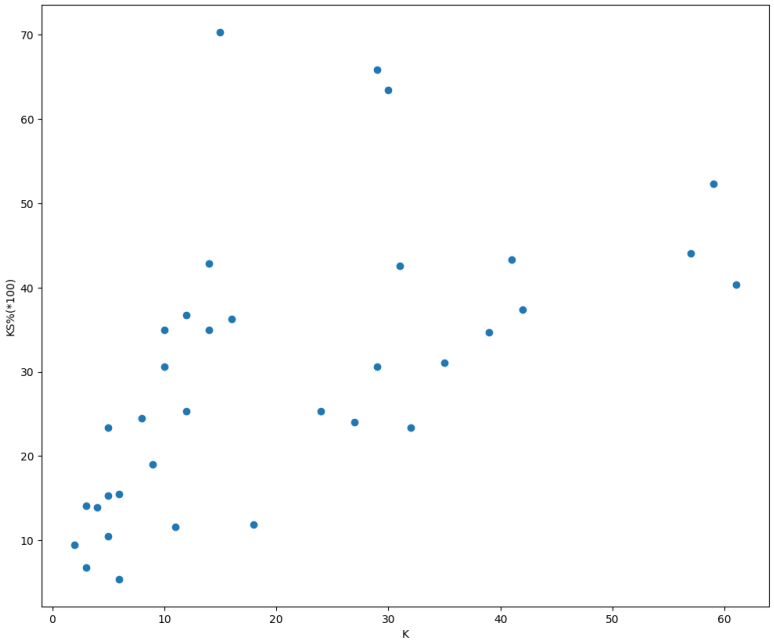
Learning to use statistical metrics such as mean, median, most common number and z-scores in my project has enabled me to better understand the distribution and characteristics of the data, leading to more accurate data analysis and decision making. The mean provides an overall trend of the data and helps me to understand the average level of the data, while the median is useful for dealing with outliers and gives a more reliable picture of where the centre of the data is. The most common number gives me an idea of the frequency distribution of the data, while the z-score is a standardised method that helps me assess how far the data points deviate from the mean.

Week 5 – Regression

This week we are interested in relationships between variables.

Firstly we collected a dataset called [LPL 2018 Summer Playoffs - Player Stats](https://oracleselixir.com/stats/players/byTournament/LPL%2F2018%20Season%2FSummer% 20Playoffs) dataset. The dataset is a collection of match data on different players in LOL esports, which includes kills, assists, vision prevalence, etc., and I was curious about what data varied widely from player to player in esports and why, and what impact it had on the kill share.

We analysed kills and kill share in the same chart using scatterplots.



Correlation

We can see from the graph that K and KS% are trending positively and we can quantitatively observe their correlation through Pearson's r coeffient. Their correlation is about 0.57, which is showing moderate correlation.

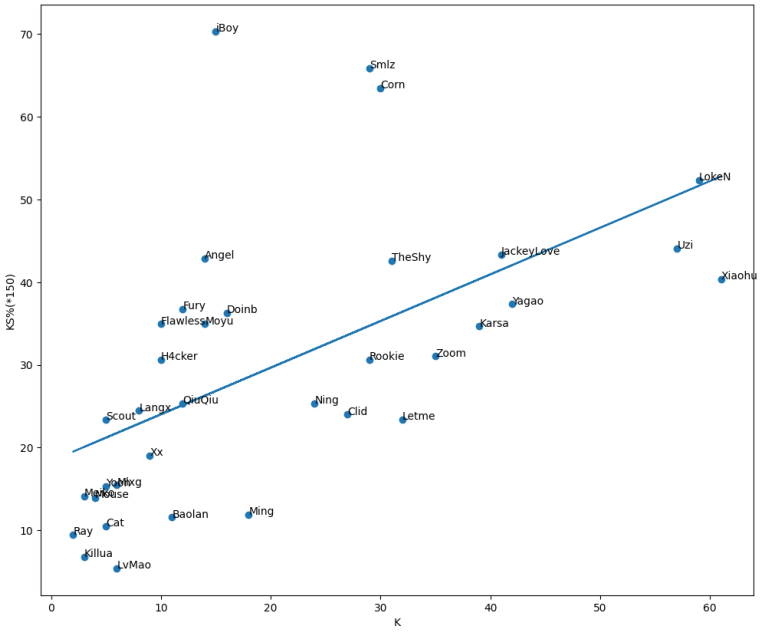
```
#Correlate
from scipy import stats
r, p = stats.pearsonr(x,y)
r
```

✓ 0.0s

```
0.5717199283278659
```

Making model

We can try to explain and predict the effect of differences in K on KS% by first building a simple statistical model.



We use r^2 (coefficient of determination) to assess the fit of our model, we can calculate the coefficient of determination as 0.37, which means that the model is not perfect, there is still 2/3 of the variability that can't be explained by the model, which could be due to receiving other factors.

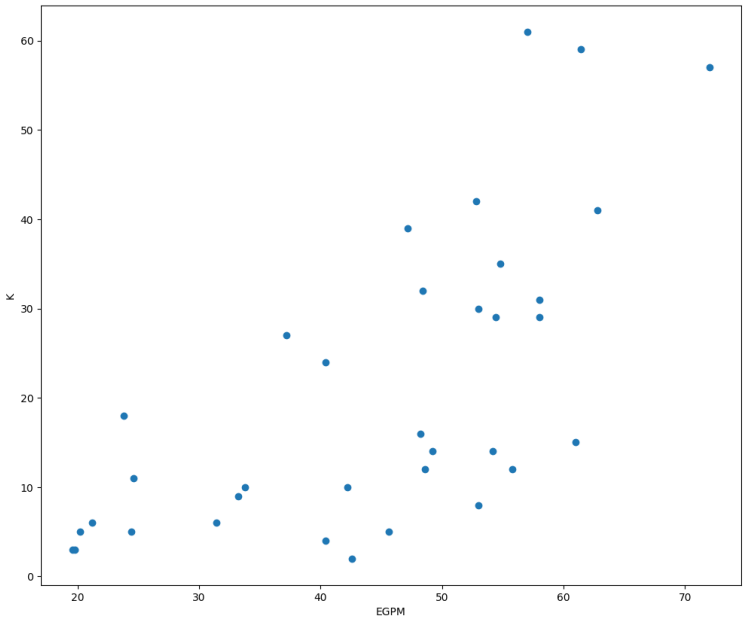
We can can determine if KS% and K are related by the p-value, we can calculate the p-value to be 0.00033, which is less than 0.05, and we can conclude that they do have a relationship.

We see that [iBoy] appears quite far above the line. From this we can determine that iBoy's KS% is more than 8 times what our model predicts. Of course this model only takes into account K, and there may be other variables that are not taken into account. But this simple linear model does seem to explain most of the difference in KS%, so players who are significantly higher than predicted may need to be investigated further, such as investigating EGPM(Average earned gold per

Player	ratioToModel
Ray	8.37231
Meiko	8.32801
iBoy	8.31029
Scout	8.29257
Flawless	6.19285

minute) to explain the difference in win rates. We can also see that proportionally, Ray has the most ratioToModel value.

We account for the presence of errors between certain observations and the model by introducing the EGPM.



We can find a relationship between K and KS% as well as a relationship between EGPM and KS%. I will be looking to see if there is a relationship between K and EGPM, as the relationship between K and KS% could be caused by the effect of EGPM on K.

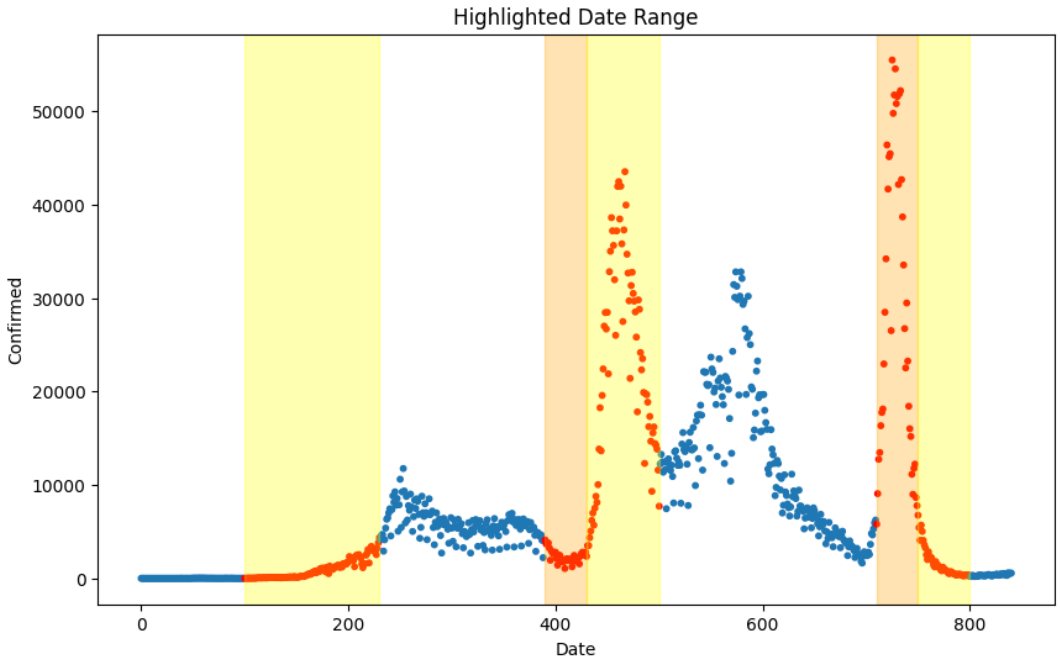
By looking at the data and fitting the model the r-score is close to 0.8 which shows that there is a strong correlation between K and EGPM. We can conclude that the relationship between K and KS% is likely due to the effect of EGPM on K.

Polynomial Regression

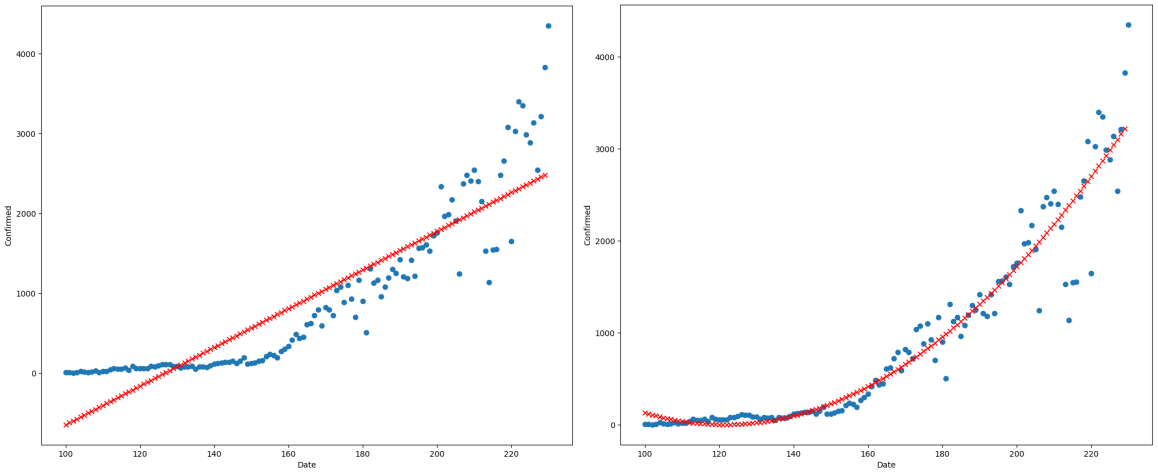
Sometimes just using one **independent** variable to predict another fails to capturethe whole relationship, so we can build a model that considers multiple variables.

We will introduce data on the [number of confirmed Covid-19 infections from 2020 to 2022](available: https://www.kaggle.com/code/stpeteishii/covid19-kerala-prediction-pycaret/input)for analysis.

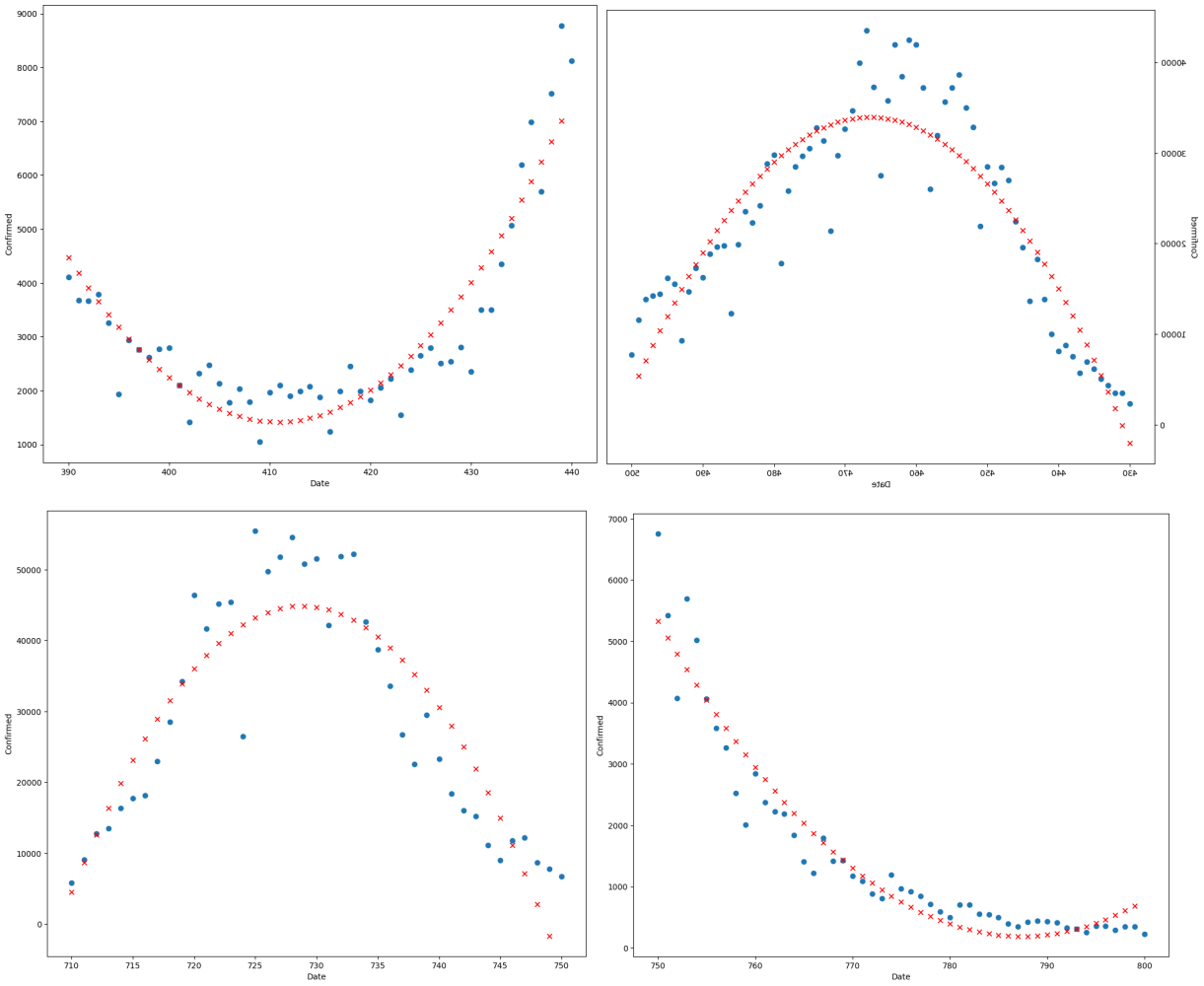
We split the data into several parts because the overall data is complex with many and complicated influencing factors, but the local data such as the period of high growth in the number of confirmed infections has fewer main influencing factors and can be analysed by polynomial regression. We can mainly divide it into rapid growth period, rebound period, peak period and decline period.



I analysed the polynomial regression for each component and found that the polynomial model fits better compared to the monomial regression.



r2 = 0.80 for monomials and r2 = 0.92 for polynomials.



Summary

Learning to use statistical metrics such as mean, median, most common number and z-scores in my project has enabled me to better understand the distribution and characteristics of the data, leading to more accurate data analysis and decision making. The mean provides an overall trend of the data and helps me to understand the average level of the data, while the median is useful for dealing with outliers and gives a more reliable picture of where the centre of the data is. The most common number gives me an idea of the frequency distribution of the data, while the z-score is a standardised method that helps me assess how far the data points deviate from the mean.

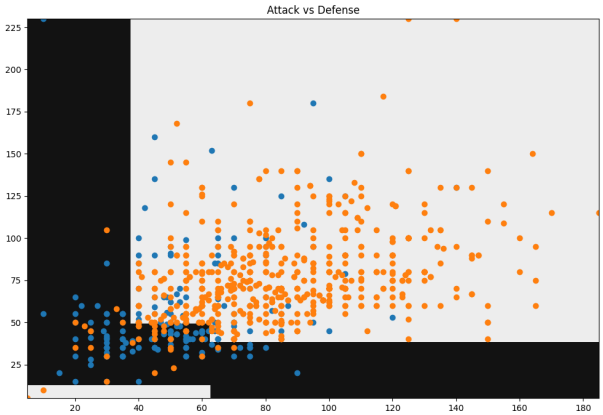
Week 6 – Classification Walkthroughs

This week we will use a parameter-free supervised learning algorithm to classify our data.

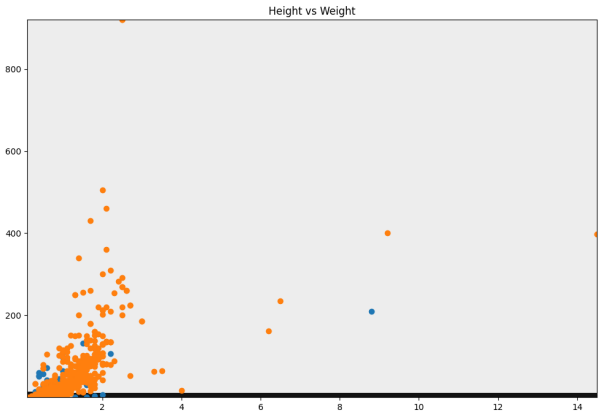
Firstly we collected a dataset called [pokedex] dataset. The dataset is a collection of different Pokémon characteristics, such as attack power, defence, life value, etc. We can try to classify the Pokémon by their height, weight, attack power, defence power, and Pokémon Life Value.

For better calculation, I converted Pokémon's life value into a binary data, the Pokémon with greater than or equal to 50 life value is a category, and the Pokémon with less than 50 life value is a category.

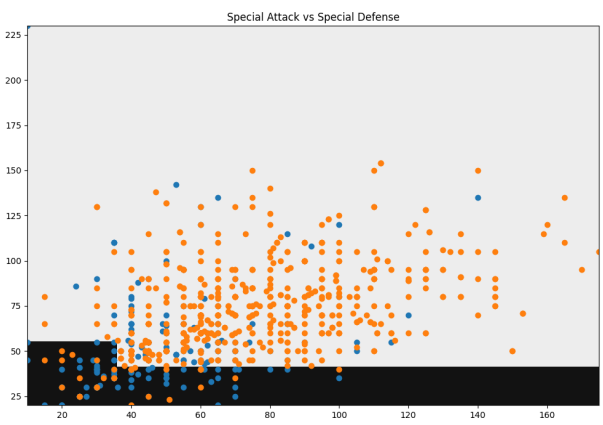
I try to classify Pokémon with different hp by decision tree classifier by drawing decision boundaries in feature space, and try to classify Pokémon by different features, different decision depths, and compare the accuracy between them.



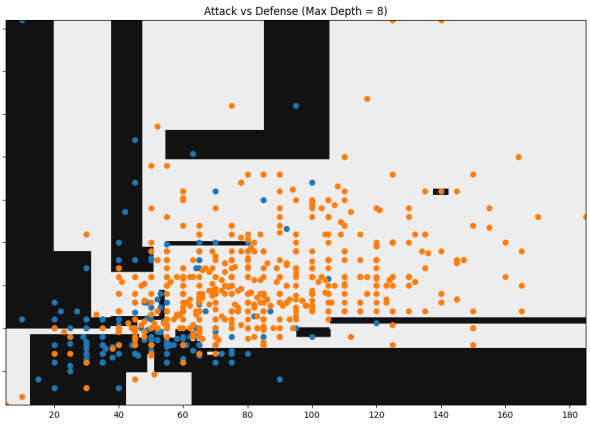
Accuracy:85.0



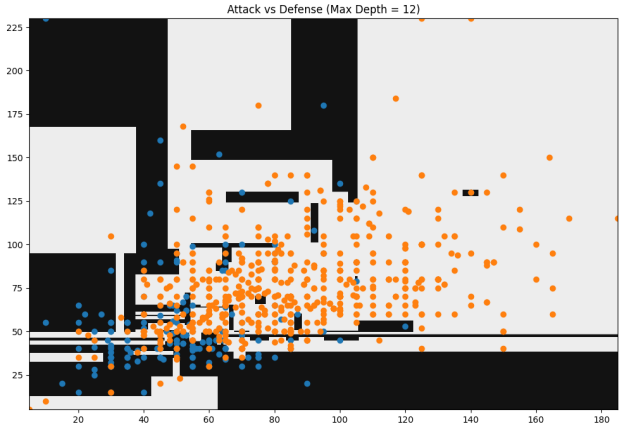
Accuracy:81.75



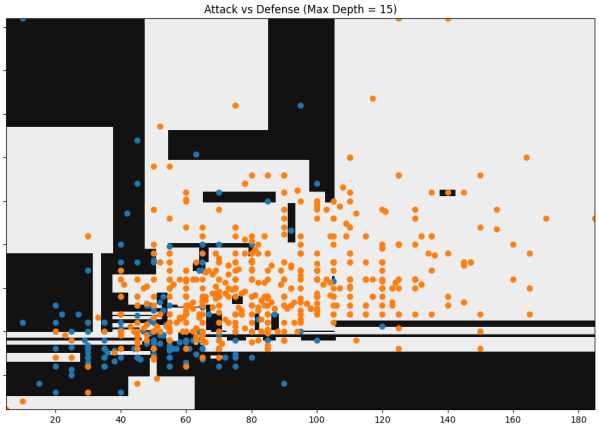
Accuracy:76.0



Accuracy:81.0



Accuracy:76.5

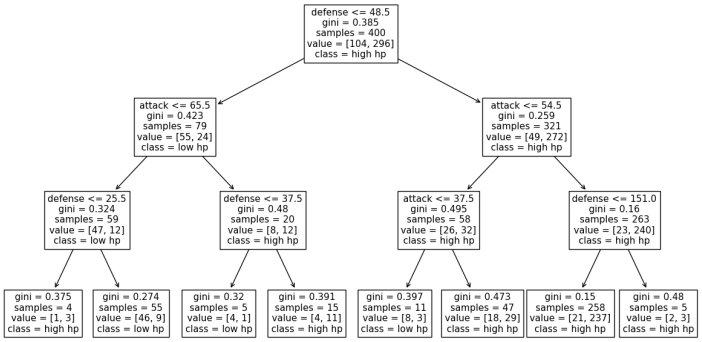


Accuracy:76.75

We can see from the accuracy that classifying a Pokémon's hp by attack and defence is the most accurate, and that increasing Max_Dept does not increase accuracy, but rather decreases it.

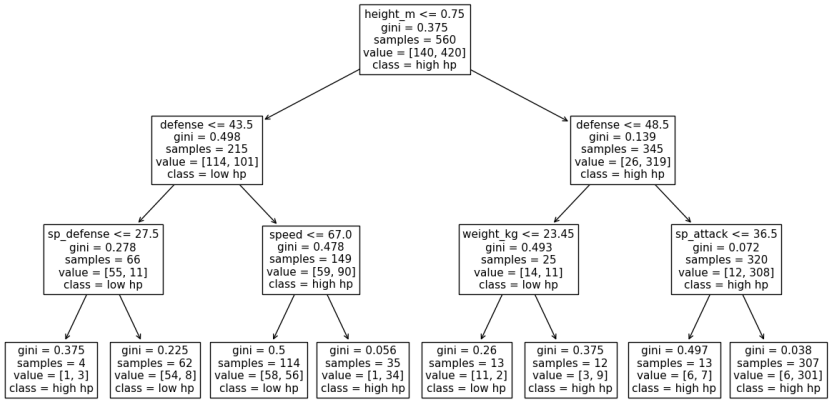
Inspecting the tree

We can see what a decision tree with the best feature pairs looks like and try to see how it has balanced both large splits and low gini impurity scores by way of images.



We can see that the first level of classification is segmented by the value of defence, the second level is segmented by the value of attack and the third level is segmented mainly by the value of defence.

We again try to use all the features for input and build the inspecting tree.



Summary

This week's classification tutorial focuses on classification tasks using a parameter-free supervised learning algorithm. Predicting HP by using traits such as Attack and Defence was tried, as well as by Sp.Attack and Sp.Defence, Height and Weight, but none of them worked as well as Attack and Defence. And I could graphically understand how machine learning algorithms learn patterns from data. In addition, I encountered the challenge of evaluating model performance and dealing with overfitting. Metrics such as accuracy and precision were examined through cross-validation and checking.

Week 8 – k-means and UMAP

Text using k-means

In this notebook, we will use the [LPL 2018 Summer Playoffs - Player Stats](https://oracleselixir.com/stats/players/byTournament/LPL%2F2018%20Season%2FSummer% 20Playoffs) dataset to try the k-means clustering algorithm. We can then examine the clusters to see if they look reasonable.

Correlation Matrix

Use the 'df.corr()' function to quickly see the correlation between features.

	GP	W%	CTR%	K	D	A	KDA	KP	KS%	DTH%	FB%	GD10	XPD10
GP	1.000000	0.558310	-0.021707	0.727997	0.857084	0.890593	0.251223	0.133805	0.042641	-0.089078	-0.188197	0.163581	0.082882
W%	0.558310	1.000000	0.279093	0.566697	0.372152	0.632773	0.698659	0.237501	0.118091	0.258549	-0.069664	0.323517	0.210238
CTR%	-0.021707	0.279093	1.000000	-0.006696	0.003141	-0.004238	0.075971	0.093961	-0.003122	0.420375	0.052941	0.102161	0.178651
K	0.727997	0.566697	-0.006696	1.000000	0.522494	0.592430	0.506043	0.166672	0.511315	-0.217392	-0.104068	0.253545	0.128253
D	0.857084	0.372152	0.003141	0.522494	1.000000	0.811648	-0.102474	0.051232	-0.088489	0.138615	-0.136589	0.076332	0.071974
A	0.890593	0.632773	-0.004238	0.592430	0.811648	1.000000	0.328272	0.192270	-0.148769	0.005191	-0.111533	0.216041	0.100732
KDA	0.251223	0.698659	0.075971	0.506043	-0.102474	0.328272	1.000000	0.418362	0.375108	-0.196402	-0.051318	0.375889	0.208871
KP	0.133805	0.237501	0.093961	0.166672	0.051232	0.192270	0.418362	1.000000	0.315310	0.030880	0.255362	0.261823	0.282635
KS%	0.042641	0.118091	-0.003122	0.511315	-0.088489	-0.148769	0.375108	0.315310	1.000000	-0.229629	0.078144	0.262175	0.215567
DTH%	-0.089078	0.258549	0.420375	-0.217392	0.138615	0.005191	-0.196402	0.030880	-0.229629	1.000000	-0.033009	0.016244	0.151221
FB%	-0.188197	-0.069664	0.052941	-0.104068	-0.136589	-0.111533	-0.051318	0.255362	0.078144	-0.033009	1.000000	0.221923	0.239374
GD10	0.163581	0.323517	0.102161	0.253545	0.076332	0.216041	0.375889	0.261823	0.262175	0.016244	0.221923	1.000000	0.828217
XPD10	0.082882	0.210238	0.178651	0.128253	0.071974	0.100732	0.208871	0.282635	0.215567	0.151221	0.239374	0.828217	1.000000
CSD10	0.147614	0.194314	0.063432	0.212102	0.131288	0.138433	0.245550	0.243594	0.289343	0.017908	0.122213	0.857790	0.809441
CSPM	-0.035959	0.048740	0.061204	0.397385	-0.200454	-0.253854	0.248185	-0.164886	0.629662	-0.312725	-0.103920	-0.005867	-0.004817
CS%P15	-0.021406	0.026078	0.056715	0.397876	-0.183075	-0.241609	0.254977	-0.174272	0.622228	-0.325335	-0.200143	-0.051763	-0.053788
DPM	0.221560	0.208308	0.105826	0.634224	0.111349	0.063890	0.318434	-0.067701	0.639359	-0.298458	-0.091402	0.177962	0.138446
DMG%	0.052127	-0.009879	0.024231	0.436976	-0.085792	-0.155496	0.240734	-0.070062	0.662688	-0.358843	-0.127874	0.042646	0.061322
EGPM	0.132298	0.300677	0.088915	0.610632	-0.050860	-0.046627	0.456413	-0.011958	0.729334	-0.267243	-0.089791	0.155022	0.100583
GOLD%	-0.018855	0.046668	0.015599	0.443604	-0.184172	-0.234809	0.297265	-0.063375	0.739119	-0.347397	-0.098686	0.047141	0.032066
WPM	0.133723	0.062579	-0.082522	-0.255497	0.222978	0.302186	-0.086054	0.144710	-0.481035	0.250112	-0.142304	0.072169	0.043544
CWPM	0.083793	0.057664	-0.128164	-0.243511	0.153819	0.299074	-0.054081	0.283633	-0.442866	0.198650	0.179710	0.135569	0.071801
WCPM	-0.077302	-0.075160	0.116637	0.055720	-0.142416	-0.082479	0.168876	0.279377	0.249253	-0.130487	0.148429	0.261688	0.111131

CSPM	CS%P15	DPM	DMG%	EGPM	GOLD%	WPM	CWPM	WCPM
-0.035959	-0.021406	0.221560	0.052127	0.132298	-0.018855	0.133723	0.083793	-0.077302
0.048740	0.026078	0.208308	-0.009879	0.300677	0.046668	0.062579	0.057664	-0.075160
0.061204	0.056715	0.105826	0.024231	0.088915	0.015599	-0.082522	-0.128164	0.116637
0.397385	0.397876	0.634224	0.436976	0.610632	0.443604	-0.255497	-0.243511	0.055720
-0.200454	-0.183075	0.111349	-0.085792	-0.050860	-0.184172	0.222978	0.153819	-0.142416
-0.253854	-0.241609	0.063890	-0.155496	-0.046627	-0.234809	0.302186	0.299074	-0.082479
0.248185	0.254977	0.318434	0.240734	0.456413	0.297265	-0.086054	-0.054081	0.168876
-0.164886	-0.174272	-0.067701	-0.070062	-0.011958	-0.063375	0.144710	0.283633	0.279377
0.629662	0.622228	0.639359	0.662688	0.729334	0.739119	-0.481035	-0.442866	0.249253
-0.312725	-0.325335	-0.298458	-0.358843	-0.267243	-0.347397	0.250112	0.198650	-0.130487
-0.103920	-0.200143	-0.091402	-0.127874	-0.089791	-0.098686	-0.142304	0.179710	0.148429
-0.005867	-0.051763	0.177962	0.042646	0.155022	0.047141	0.072169	0.135569	0.261688
-0.004817	-0.053788	0.138446	0.061322	0.100583	0.032066	0.043544	0.071801	0.111131
-0.011762	-0.050082	0.177662	0.086228	0.131328	0.049195	0.081168	0.125597	0.187508
1.000000	0.982428	0.840332	0.899245	0.939477	0.978265	-0.798709	-0.875856	-0.055602
0.982428	1.000000	0.843167	0.911389	0.923192	0.966752	-0.727169	-0.876042	-0.039062
0.840332	0.843167	1.000000	0.923824	0.900588	0.854447	-0.595251	-0.685286	0.023004
0.899245	0.911389	0.923824	1.000000	0.869333	0.916179	-0.646362	-0.768211	0.025797
0.939477	0.923192	0.900588	0.869333	1.000000	0.956094	-0.735421	-0.782821	0.029567
0.978265	0.966752	0.854447	0.916179	0.956094	1.000000	-0.797510	-0.850147	0.051807
-0.798709	-0.727169	-0.595251	-0.646362	-0.735421	-0.797510	1.000000	0.789070	-0.136754
-0.875856	-0.876042	-0.685286	-0.768211	-0.782821	-0.850147	0.789070	1.000000	0.146578
-0.055602	-0.039062	0.023004	0.025797	0.029567	0.051807	-0.136754	0.146578	1.000000

We can see that a small portion of the data is extremely correlated, but this is probably due to the fact that the data are all derived from the same type of data (e.g. damage per minute and damage percentage).

Process data

In order to improve the clustering results, we prepared the data, analysed the clusters, visualised the clusters, determined the number of clusters using the 'elbow method', re-dimensioned and re-clustered the data, and analysed the clustering results prior to performing the cluster analysis.

The text data was preprocessed and features were extracted and then the text data was analysed for clustering using the K-means algorithm. By visualising the clustering results, we can observe the distribution and similarity between different clusters. The most appropriate number of clusters is determined to avoid overfitting or underfitting. Subsequently, based on the determined optimal number of clusters, the text data is subjected to dimensionality reduction.

After the above data processing, we can use K-means algorithm for clustering and analysis.

Cluster Results

	Player	Team	Pos	GP	W%	CTR%	K	D	A	KDA
1	Alex	FunPlus Phoenix	Jungle	3	0.33	0.67	4	7	16	2.9
5	Clearlove	EDward Gaming	Jungle	2	0.00	0.00	3	3	4	2.3
12	Flawless	Rogue Warriors	Jungle	4	0.25	0.50	10	14	22	2.3
16	Haro	EDward Gaming	Jungle	2	0.50	0.50	4	4	8	3.0
22	Kiwi	Rogue Warriors	Jungle	3	0.00	0.33	1	7	14	2.1
31	Mlxg	Royal Never Give Up	Jungle	4	0.50	1.00	6	12	32	3.2
43	Xx	Top Esports	Jungle	8	0.38	0.38	9	19	46	2.9

Cluster:0

	Player	Team	Pos	GP	W%	CTR%	K	D	A	KDA
3	Baolan	Invictus Gaming	Support	10	0.50	0.50	11	30	90	3.4
4	Cat	Top Esports	Support	8	0.38	1.00	5	25	34	1.6
9	Crisp	FunPlus Phoenix	Support	3	0.33	0.33	4	5	17	4.2
21	Killua	Rogue Warriors	Support	7	0.14	0.43	3	29	44	1.6
27	LvMao	JD Gaming	Support	15	0.67	0.33	6	38	114	3.2
29	Meiko	EDward Gaming	Support	4	0.25	0.75	3	12	21	2.0
30	Ming	Royal Never Give Up	Support	13	0.69	0.54	18	31	134	4.9
45	Yoon	Suning	Support	4	0.50	0.00	5	10	31	3.6

Cluster:1

	Player	Team	Pos	GP	W%	CTR%	K	D	A	KDA
8	Corn	Top Esports	Middle	8	0.38	0.50	30	21	20	2.4
10	Doinb	Rogue Warriors	Middle	7	0.14	0.43	16	20	30	2.3
13	Fury	Suning	ADC	4	0.50	1.00	12	7	23	5.0
18	iBoy	EDward Gaming	ADC	4	0.25	0.25	15	5	11	5.2
19	JackeyLove	Invictus Gaming	ADC	10	0.50	0.40	41	27	51	3.4
23	Langx	Suning	Top	4	0.50	1.00	8	10	21	2.9
26	LokeN	JD Gaming	ADC	15	0.67	0.73	59	17	65	7.3
28	Lwx	FunPlus Phoenix	ADC	3	0.33	0.33	10	3	9	6.3
37	Rookie	Invictus Gaming	Middle	10	0.50	0.70	29	25	76	4.2
39	Smlz	Rogue Warriors	ADC	7	0.14	0.29	29	25	20	2.0
40	TheShy	Invictus Gaming	Top	7	0.57	0.57	31	26	37	2.6
41	Uzi	Royal Never Give Up	ADC	10	0.80	0.50	57	16	69	7.9
42	Xiaohu	Royal Never Give Up	Middle	13	0.69	0.31	61	32	101	5.1
44	Yagao	JD Gaming	Middle	15	0.67	0.47	42	25	76	4.7

Cluster:2

	Player	Team	Pos	GP	W%	CTR%	K	D	A	KDA
0	Able	Royal Never Give Up	ADC	3	0.33	1.00	9	9	12	2.3
2	Angel	Suning	Middle	4	0.50	0.50	14	8	15	3.6
7	Cool	FunPlus Phoenix	Middle	3	0.33	0.67	7	8	15	2.8
11	Duke	Invictus Gaming	Top	3	0.33	0.33	6	4	10	4.0
14	GimGoon	FunPlus Phoenix	Top	3	0.33	0.33	0	5	13	2.6
17	Holder	Rogue Warriors	Top	3	0.00	0.67	3	12	5	0.7
25	Lies	Top Esports	Top	2	0.00	0.00	1	5	2	0.6
32	Mouse	Rogue Warriors	Top	4	0.25	0.75	4	15	22	1.7
33	Moyu	Top Esports	Top	6	0.50	0.33	14	11	28	3.8
35	QiuQiu	Top Esports	ADC	8	0.38	0.13	12	28	24	1.3
36	Ray	EDward Gaming	Top	4	0.25	0.00	2	5	20	4.4
38	Scout	EDward Gaming	Middle	4	0.25	0.75	5	7	18	3.3

Cluster:3

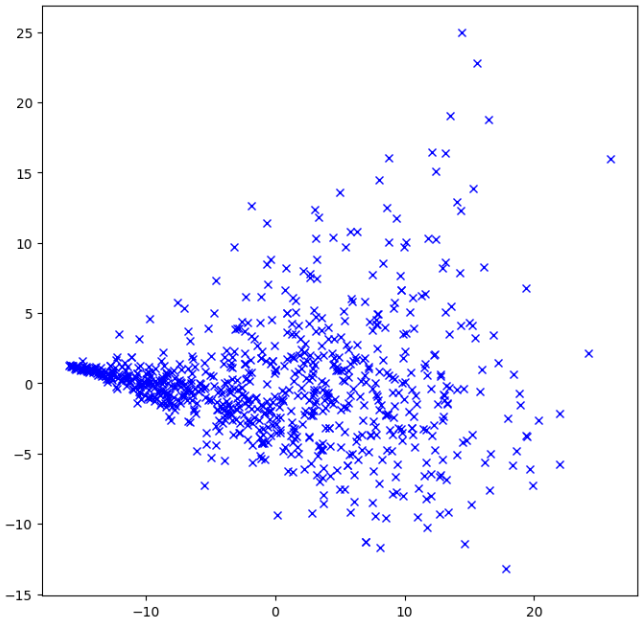
	Player	Team	Pos	GP	W%	CTR%	K	D	A	KDA
6	Clid	JD Gaming	Jungle	15	0.67	0.53	27	33	84	3.4
15	H4cker	Suning	Jungle	4	0.50	0.50	10	7	28	5.4
20	Karsa	Royal Never Give Up	Jungle	9	0.78	0.33	39	22	88	5.8
24	Letme	Royal Never Give Up	Top	12	0.67	0.42	32	28	82	4.1
34	Ning	Invictus Gaming	Jungle	10	0.50	0.60	24	38	73	2.6
46	Zoom	JD Gaming	Top	15	0.67	0.60	35	30	79	3.8
47	Zz1tai	Royal Never Give Up	Top	1	1.00	1.00	5	2	10	7.5

Cluster:4

We can clearly notice from the clustered data with different results that there are two clusters in which all the players have a 'POS' of 'Junngle' or all of them are 'Support'; and in another cluster most of his players are 'Top'; in another cluster all the players in it have a win rate much higher than 50 per cent; and in the last cluster all the players in it have a high value of 'K': this suggests that the effect of the clustering has a significant effect.

Image using k-means

In this notebook, we will try the k-means clustering algorithm using the [OSCA](https://www.pinterest.com/ceo_of_bennefischl/asuka-is-just-like-me/) image. Then, we can check if the clustering looks reasonable. Unlike textual data which can be processed using the K-means algorithm, textual data can usually be converted into a numerical feature representation, whereas image data has to be analysed by obtaining features from its pixel information and converting it into a two-dimensional array.

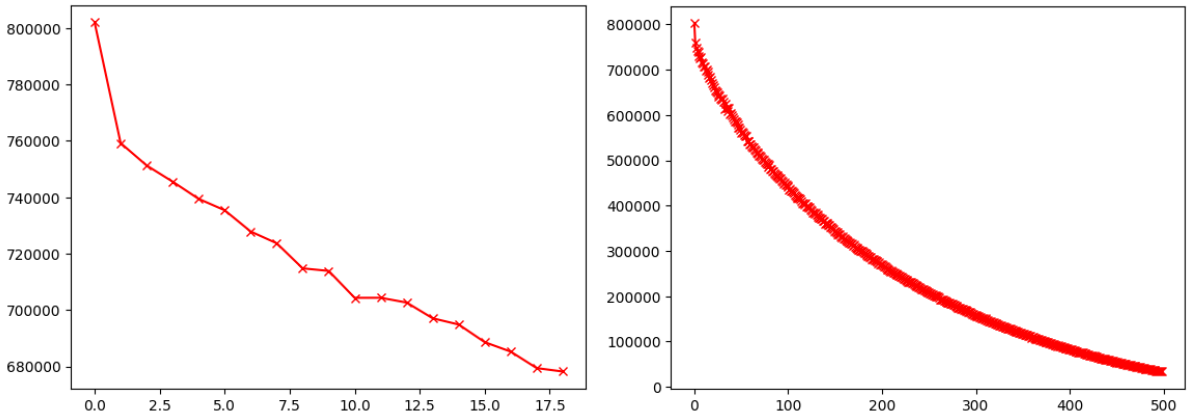


2D dataset of images plotted on 2D axes

The image data has to be characterised by its pixel information and converted into a two-dimensional array for analysis.

Elbow plot

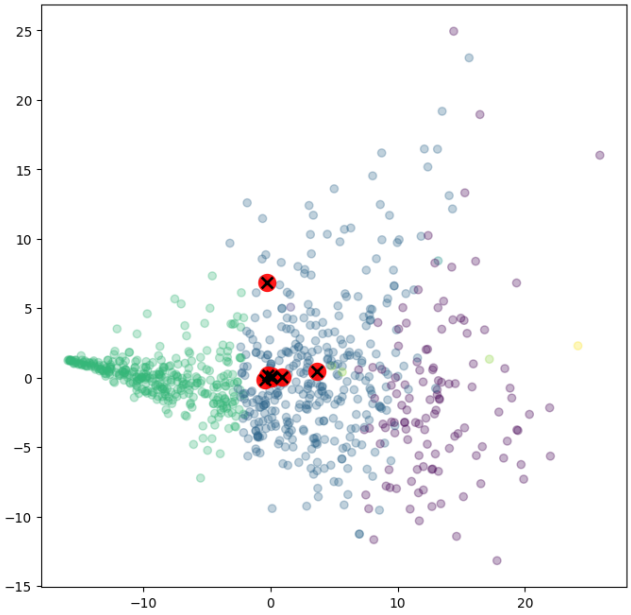
We can use elbow plot to help us determine the optimal number of clusters in K-means clustering by means of a graph.



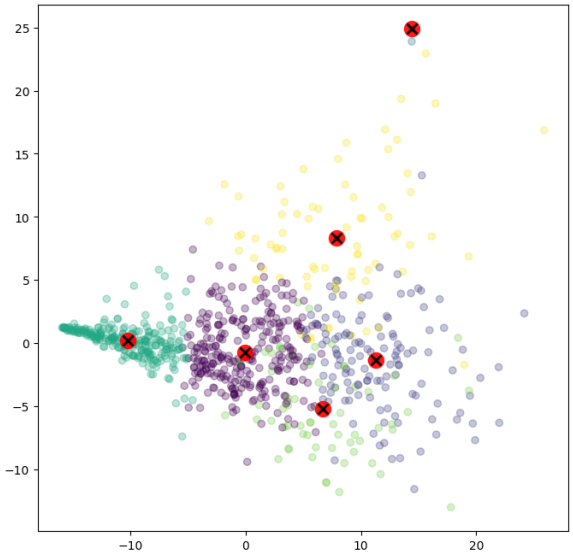
The graph on the left shows the sum of squares of the in-sample distances for 20 images, but the score is still above 680,000 and I didn't notice the decline becoming flat in the image, so I set the RANGE to (1, 500) and looked at the score change. In Figure 2 we can see that the score is approaching 0 and at the end the graph has taken on a shape similar to the bend of an arm, which we can roughly determine to be the elbow point.

Cluster and Plot

We can start by clustering all features and then reduce to 2 dimensions for plotting.



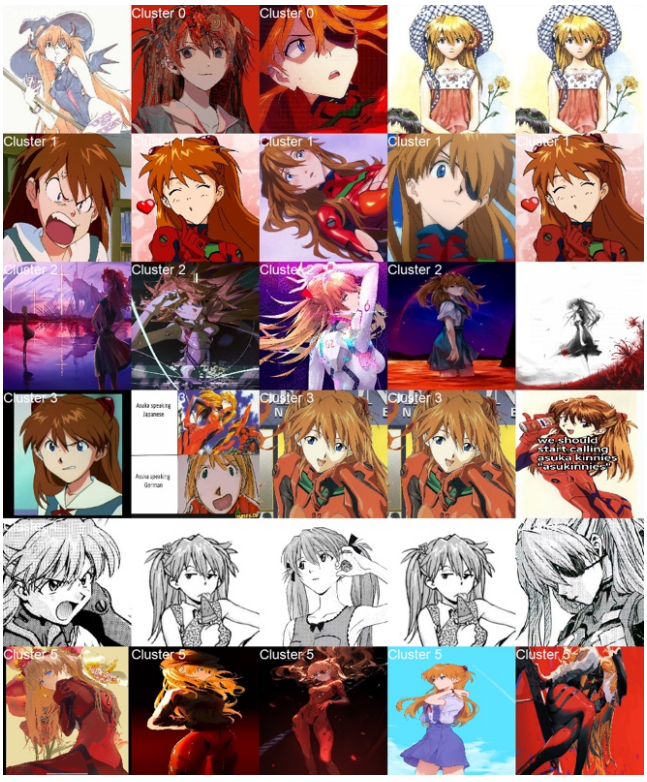
We can find that the clusters are more centralised and do not show a good ability to classify clusters. So we decided to reduce Dimensions before clustering.



We can find from the figure that this time the clusters are classified better and the main clusters are separated. And we can find two clusters with less data on top of the image, this may be because we set the classification to 7 clusters, we can merge it into one cluster, which is k=6.

Examine Clusters

After cluster classification, we can judge whether the cluster classification is successful or not by checking the representative images of each cluster, by means of images.



We can see from the picture that each group of pictures basically has a distinct style, such as Cluster4 which is a black and white comic book style. However, there are also pictures in the classification that don't quite match such as the fourth picture of Cluster5, other pictures have obvious black and red backgrounds, while it has a blue sky background. This shows that there is still room for improvement in the model.

UMAP

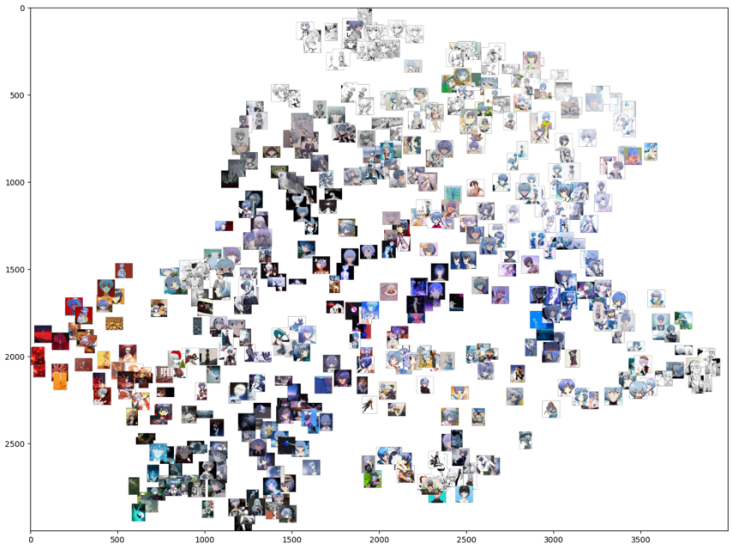
In this notebook, we will try UMAP using [Rei | Evangelion] (https://www.pinterest.com/salmon00_601/rei-evangelion/) image datasets of different kinds (manga, original drawings, emoticons).UMAP can map high dimensional data to low dimensional space while preserving the local structure of the data, which can help us to cluster the image data.

The source code is downloaded from the URL of the image obtained through the API and connected to the model. I already had a database of images in a local file, so I modified the code to reference the local file.

```
def collect_data(directory_path, max_images=500):
    # Use glob pattern matching to find all image files
    pattern = os.path.join(directory_path, '*.jpg')
    image_paths = glob.glob(pattern)
    # Limit the maximum number of images
    image_paths = image_paths[:max_images]
    if not image_paths:
        print('No images found. Check the directory path.')
        return []
    print(f'Found {len(image_paths)} images. Processing...')
    # Here we simply return the paths of the images, or you can further process as needed
    return image_paths
# usage
directory_path = 'data/Rei | Evangelion'
images = collect_data(directory_path)
print(f'Collected {len(images)} images.')
```

UMAP Visualisation 1: Colour palettes

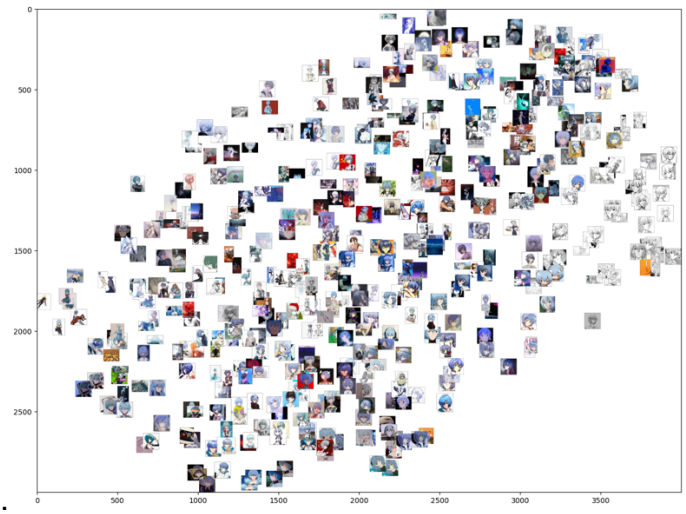
The first UMAP visualisation technique we tried was palette-based.



We can see from the diagram that the images are mostly in RGB colours have a good classification.

UMAP Visualisation 2: ImageNet Features

Next we will perform UMAP visualisation using ImageNet features to reduce the original image to 1000 semantic dimensions



From the figure, we can find that most of the pictures with convergent painting style and convergent colours are in the same area, which can assist our values to show the distribution of image features, and we can find that some of the images in our image dataset are duplicated from where the images are clustered in the figure.

Summary

Through this week, I gained insight into the application of the k-means and UMAP algorithms to data clustering and dimensionality reduction.

Firstly, I used the [LPL Summer 2018 Playoffs - Player Data] dataset to try out the k-means clustering algorithm. By observing the clustering results, I found some interesting phenomena, such as all players' positions in certain clusters are 'Jungle' or all are 'Support', which indicates that the clustering effect is obvious.

Next, I used the k-means algorithm to cluster the image data. Unlike text data, image data requires features to be extracted from the pixel information and converted into a two-

dimensional array for analysis. By plotting the elbow diagram, I determined the optimal number of clusters and observed a visualisation plot of the clustering results. Performing clustering before and after dimensionality reduction, I found that the clustering was better after dimensionality reduction and the main clusters were more clearly separated, which suggests that dimensionality reduction is important for improving clustering.

Finally, I tried dimensionality reduction and clustering on the [Rei | Evangelion] image dataset using the UMAP algorithm. By converting calls to the API into calls to locally acquire the image data and connect to the model, I successfully processed the image data and visualised it using the UMAP algorithm. I tried two UMAP visualisation techniques, a palette-based visualisation and a dimensionality reduction using ImageNet features. From the results, I found that these visualisation techniques were effective in showing the distribution and features of the image data and helped to understand the data better.

Code

All code is in the GitHub repository https://git.arts.ac.uk/23009764/intro-to-ds-portfolio

Most code is started by prompting ChatGPT then tailoring for specific use cases