Machine Learning and Password Classification

Aims

- First attempt at machine learning
- 3rd year project is using reinforcement learning to design navigation systems of the UCL Mars Rover
- Phase 1: Train model using dataset of passwords
- Phase 2: Write code that rates a password you give it, if rating is below a certain threshold, it will improve the password
- Problem: Model was not accurate enough for phase 2 to do its job
- Will explore accuracy of model using different algorithms

Layout of an ideal model

features:

- Password length
- No. of uppercase characters
- No. of lowercase characters
- No. of digits
- No. of symbols

labels:

0 – very weak password

1 – weak password

2 – good password

3 – Ideal password

What does success look like?

- The model should successfully be able to rate a password with a 1 or a 0

- Increased complexity by increasing the range of numbers to denote strength of password

- Aiming for an accuracy of 60%

Dataset A:

- 140 passwords
- Labels are as follows:
- 0 : worst passwords found online
- 1 : three words 3 integers

Easier to train models as there are clear differences between passwords

Dataset B:

- 204 passwords
- Labels are as follows:
- 0 : worst passwords found online
- 1 : one word 2 integers
- 2 : two words 3 integers
- 3 : three words 3 integers

Trained models will be able to guess passwords with different strengths

More passwords used to train to give model better metrics

Methodology

1) Import relevant modules

2) Import dataset

3) Convert text into numbers (tokens) using TF-IDF Vectorizer

4) Implement algorithm to train model

5) Make predictions of strength of password

6) Find accuracy, precision and recall by comparing predictions to correct values

ML algorithms

Supervised learning:

- Naïve Bayes
- Linear regression
- k-nearest neighbors (kNN)
- Support Vector Machines (SVM)

Unsupervised learning:

- K-means Clustering

Key words

Precision:
- Measure of quality
- High precision means model returns more relevant results than irrelevant results

Recall:
- Measure of quantity
- High recall means model returns most relevant results (regardless of whether irrelevant is also returned)

Accuracy:
- How often a classification is correct overall

Unsupervised learning: K-means clustering

Dataset A:

- Accuracy : 0.507
- Precision : 0.5036
- Recall : 1

Dataset B:

- Accuracy : 0.2549
- Precision : 0.3134
- Recall : 0.2549

- Finds similarity between items and groups them into k amounts of clusters

- Only uses input data without knowing what is or isn’t the correct answer

Naïve Bayes

Dataset A:

- Accuracy : 0.86
- Precision : 0.89
- Recall : 0.86

Dataset B:

- Accuracy : 0.56
- Precision : 0.58
- Recall : 0.56

- probabilistic algorithm based on Bayes' theorem
- models the probability of each class based on the feature values
- Assumes features are conditionally independent of labels
- Commonly used for textual data

Logistic regression

Dataset A:
- Accuracy : 0.93
- Precision : 0.88
- Recall : 1

Dataset B:
- Accuracy : 0.61
- Precision : 0.64
- Recall : 0.61

- relationship between features and the output as a linear combination
- suitable for binary and multi-class classification tasks when the decision boundary is assumed to be linear

K Nearest Neighbor

Dataset A:
- Accuracy : 0.89
- Precision : 0.9118
- Recall : 0.8929

Dataset B:
- Accuracy : 0.63
- Precision : 0.6676
- Recall : 0.6341

- Instance based algorithm : doesn't build an explicit model during training
- Makes prediction based on similarity between data points
- Suitable for both classification and regression and works well with both linear and non-linear decision boundary

K Nearest Neighbor

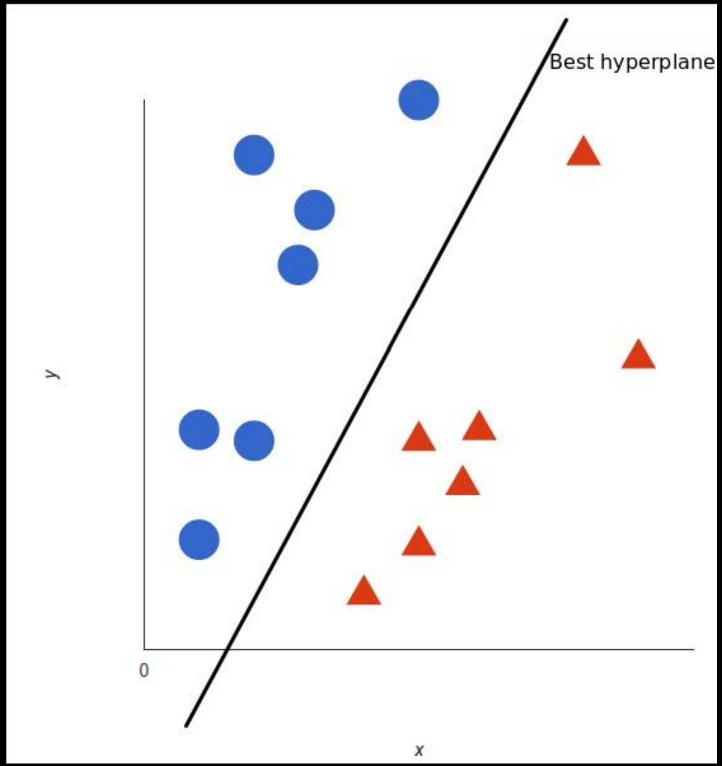
K value	Accuracy	Precision	Recall
1	0.56	0.6235	0.5610
3	0.63	0.6423	0.6341
4	0.61	0.6333	0.6098
5	0.63	0.6676	0.6341
6	0.61	0.6766	0.6098
7	0.61	0.6872	0.6098

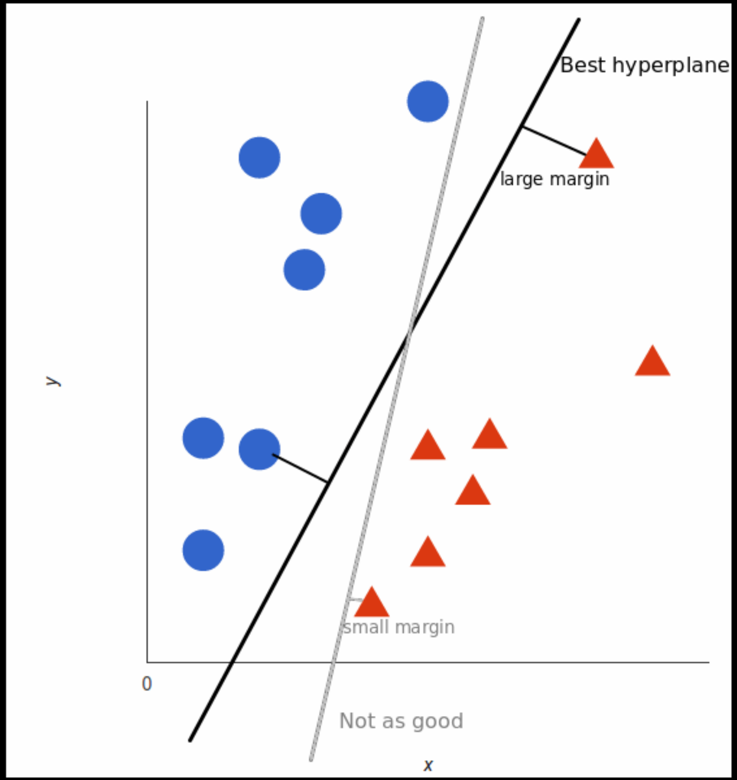
k = 5

Support Vector Machines (SVM)

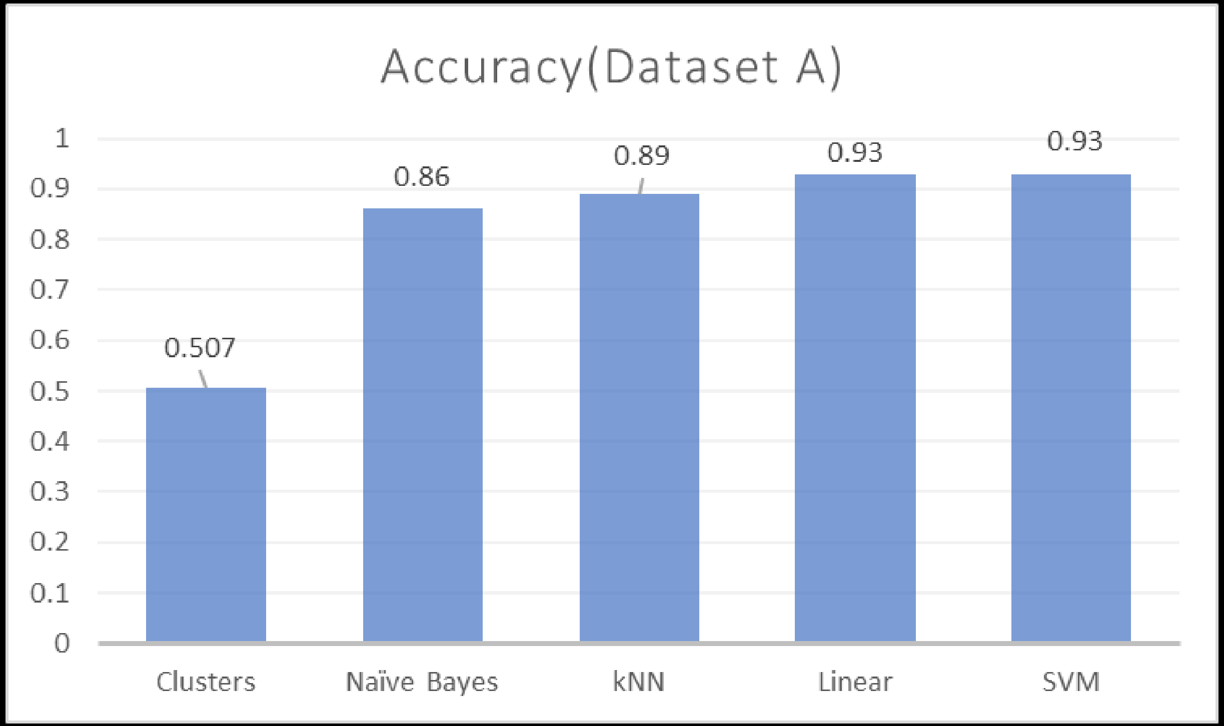
Dataset A:
- Accuracy : 0.93
- Precision : 0.94
- Recall : 0.93

Dataset B:
- Accuracy : 0.73
- Precision : 0.81
- Recall : 0.73





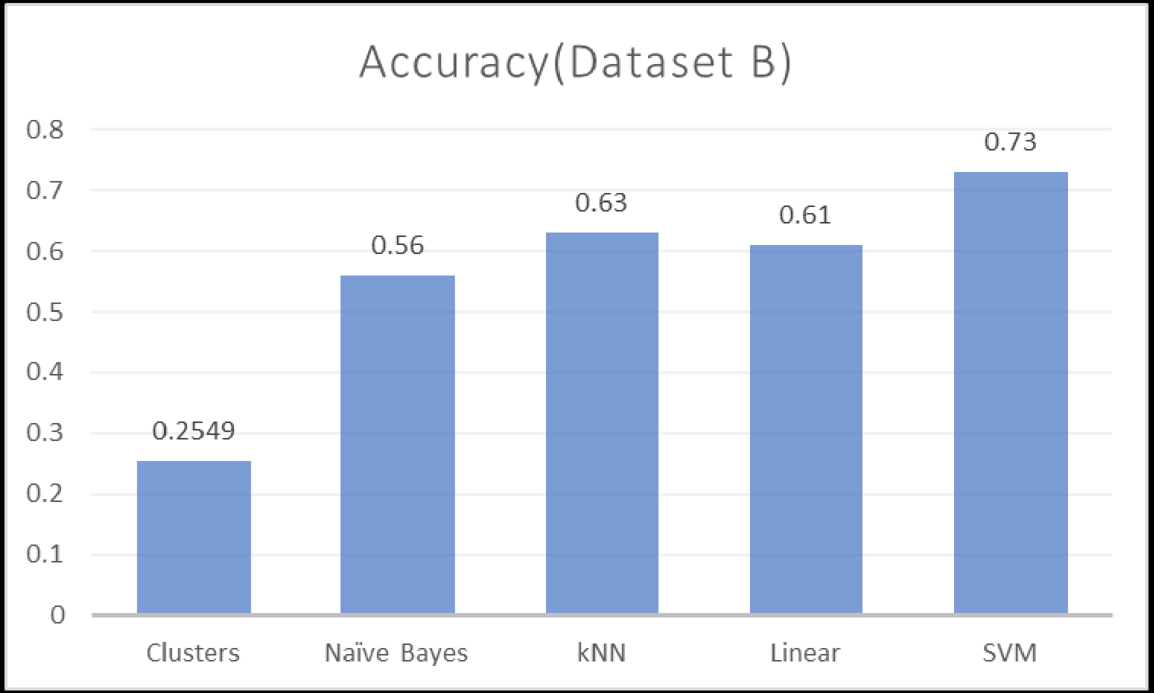
[1]



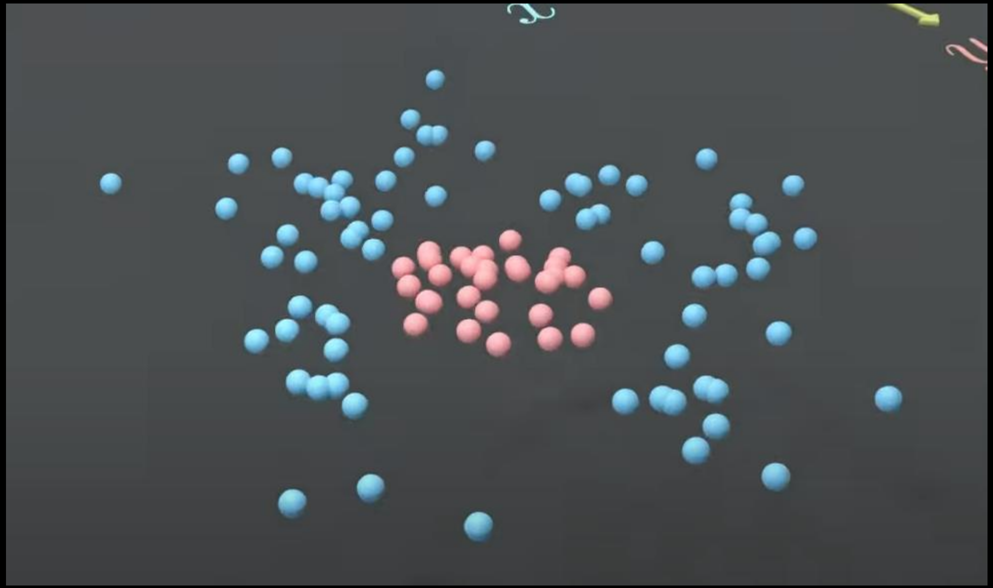
- Unsupervised has much lower accuracy rates

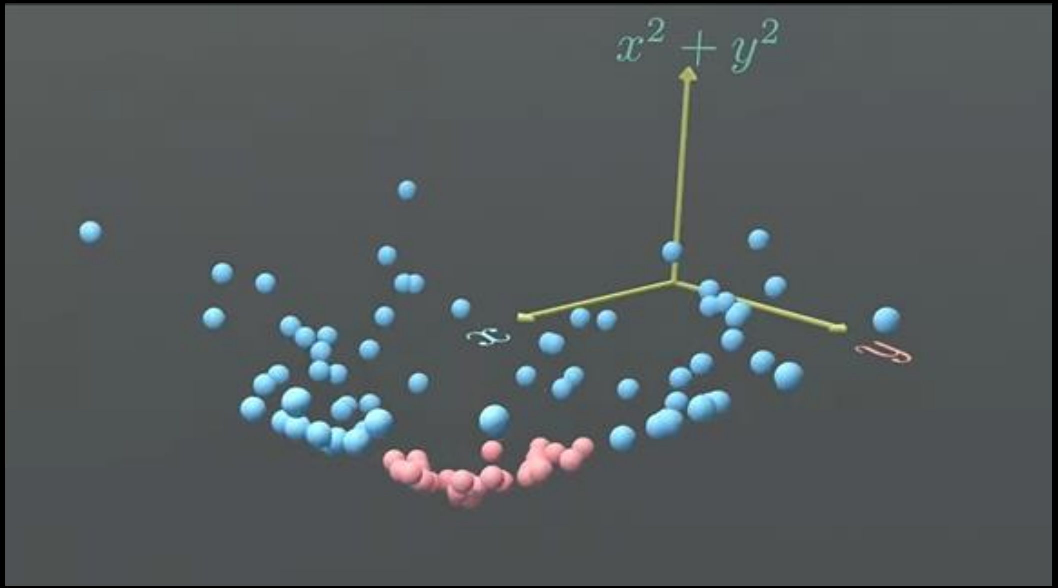
- Both Linear and SVM performed equally well due to linear nature of the data

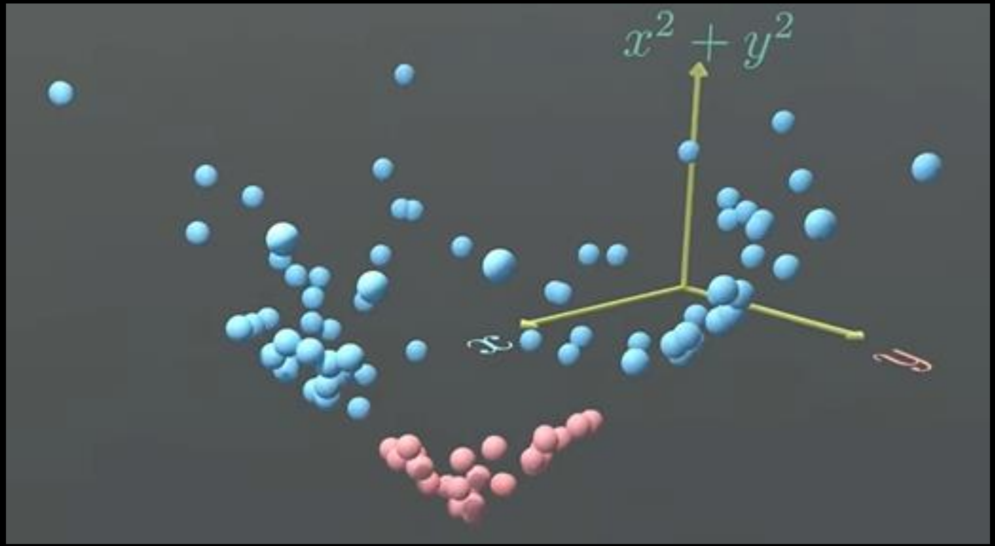
- Not much insight can be gained from this graph as differences are not large for supervised and position for more accurate dataset is a tie

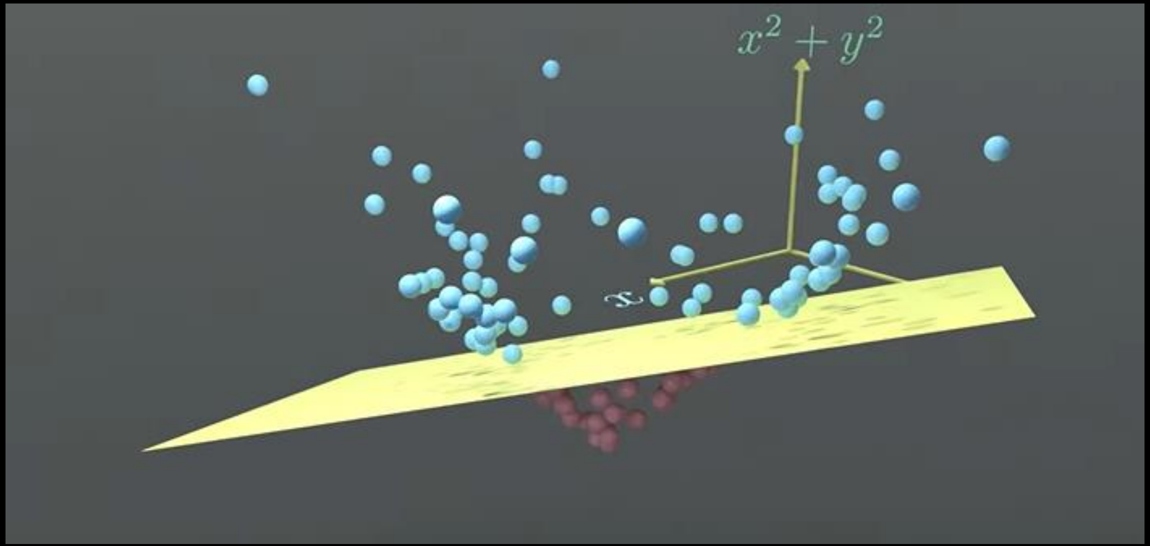


- SVM have built-in regularization which helps prevent over-fitting
- kNN and Naïve Bayes don’t have inherent regularization mechanisms
- SVM have ability to handle high dimensional data and are robust to irrelevant features
- kNN outperformed Linear because its possible that passwords had non-linearity
- SVM outperformed both as it is equipped to handle both linearity and non-linearity









[2]

What I would do differently

- Larger dataset of words to generate good and bad passwords
- Attempt more complex variation of every algorithm
- Implement password generation for algorithm with highest accuracy

Thank you!

Complications

- CountVectorizer only works with strings so had to treat integers in passwords as strings

- Issues with the dataset – to generate a list of strong passwords, I used a limited number of words so when model sees a password which has words it doesn't recognize from past passwords, then it assigns the password a 0, classing the password as very weak even if it might not be

- 'ValueError: Target is multiclass but average = "binary"' – when calculating precision/recall, the gradings for passwords were non-binary so I had to change the code so that average = "weighted"

References

[1] MonkeyLearn Blog. (2017). An Introduction to Support Vector Machines (SVM). [online] Available at: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/#:~:text=A%20support%20vector%20machine%20(SVM.

[2] www.youtube.com. (n.d.). Support Vector Machine (SVM) in 2 minutes. [online] Available at: https://www.youtube.com/watch?v=_YPScrckx28