Data Set Description

1) S1 “Suit of card #1”
Nominal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

2) C1 “Rank of card #1”
Ordinal (1-13) representing {Ace, 2, 3, …, Queen, King}

3) S2 “Suit of card #2”
Nominal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

4) C2 “Rank of card #2”
Ordinal (1-13) representing {Ace, 2, 3, …, Queen, King}

5) S3 “Suit of card #3”
Nominal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

6) C3 “Rank of card #3”
Ordinal (1-13) representing {Ace, 2, 3, …, Queen, King}

7) S4 “Suit of card #4”
Nominal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

8) C4 “Rank of card #4”
Ordinal (1-13) representing {Ace, 2, 3, …, Queen, King}

9) S5 “Suit of card #5”
Nominal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

10) C5 “Rank of card #5”
Ordinal (1-13) representing {Ace, 2, 3, …, Queen, King}

11) Class “Poker Hand”
Ordinal (0 - 9)

Data Set Description

Output: Class

"Poker Hand" Ordinal (0 - 9)

0: Nothing in hand; not a recognized poker hand
1: One pair; one pair of equal ranks within five cards
2: Two pairs; two pairs of equal ranks within five cards
3: Three of a kind; three equal ranks within five cards
4: Straight; five cards, sequentially ranked with no gaps
5: Flush; five cards with the same suit
6: Full house; pair + different rank three of a kind
7: Four of a kind; four equal ranks within five cards
8: Straight flush; straight + flush
9: Royal flush; {Ace, King, Queen, Jack, Ten} + flush

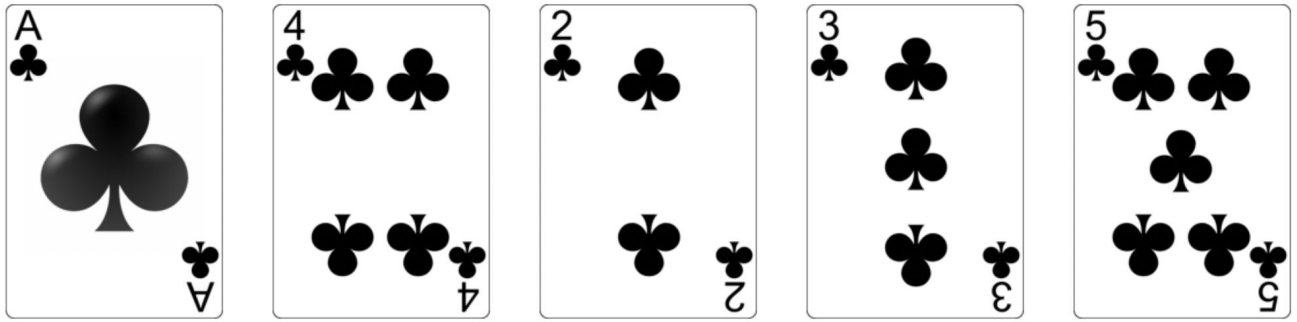
Data Set Description

Input labels: [‘S1’, ‘C1’, ‘S2’, ‘C2’, ‘S3’, ‘C3’, ‘S4’, ‘C4’, ‘S5’, ‘C5’]

Output label: [‘Class’]

S1	C1	S2	C2	S3	C3	S4	C4	S5	C5	Class
4	1	4	4	4	2	4	3	4	5	8

Straight flush



Number of training points: 25,000

Number of test points: 100,000

Data Preprocessing

Nominal:

Nominal data is defined as data that is used for naming or labelling variables, without any quantitative value.

There is usually no intrinsic ordering to nominal data.

Input labels: [‘S1’, ‘S2’, ‘S3’, ‘S4’, ‘S5’]

	C1	C2	C3	C4	C5	Class	S1_1	S1_2	S1_3	S1_4	...	S3_3	S3_4	S4_1	S4_2	S4_3	S4_4	S5_1	S5_2	S5_3	S5_4
0	1	13	4	3	12	0	1	0	0	0	...	0	0	0	1	0	0	1	0	0	0
1	12	2	11	5	5	1	0	0	1	0	...	1	0	0	0	0	1	0	1	0	0
2	9	6	4	2	9	1	1	0	0	0	...	0	0	0	0	1	0	0	0	1	0
3	4	13	13	1	6	1	1	0	0	0	...	0	0	0	1	0	0	0	0	1	0
4	10	7	2	11	9	0	0	0	1	0	...	0	0	0	1	0	0	0	0	0	1

Ordinal:

The variables in ordinal data are listed in an ordered manner.

The ordinal variables are usually numbered, so as to indicate the order of the list.

Input labels: [‘C1’, ‘C2’, ‘C3’, ‘C4’, ‘C5’]

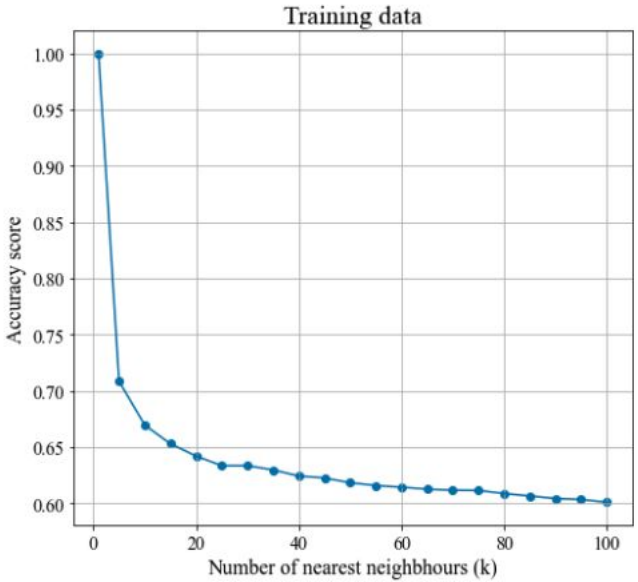
Note: Dummy variables of Nominal variables are created before fitting the model.

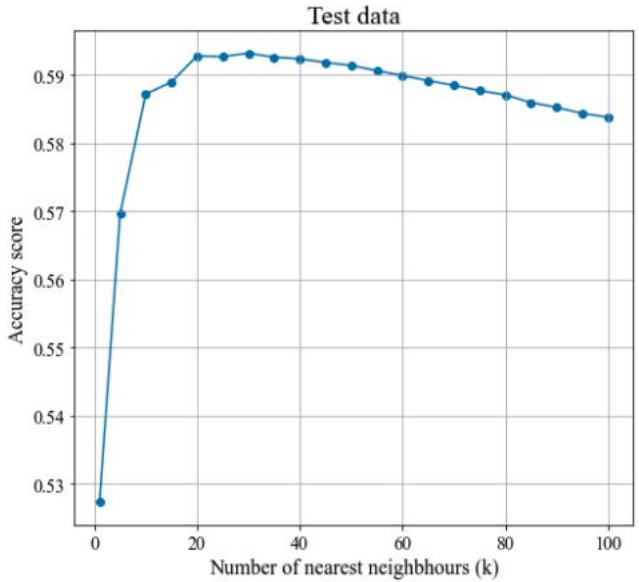
K-Nearest Neighbours

Parameter Tunning

Search Space: {1, 5, 10, …, 100}

- Maximum accuracy on test dataset is obtain for k=30.

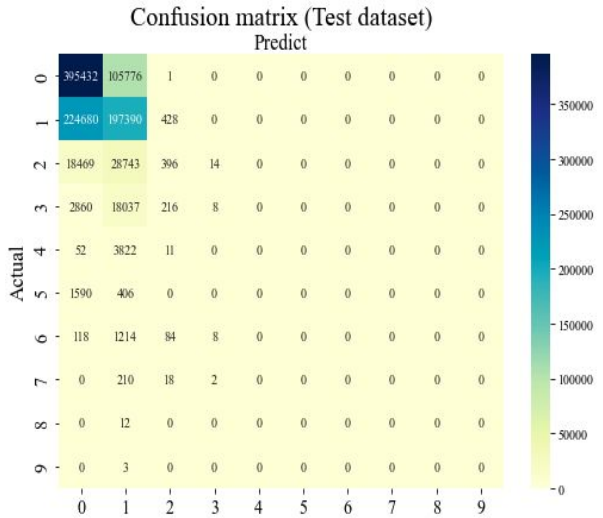


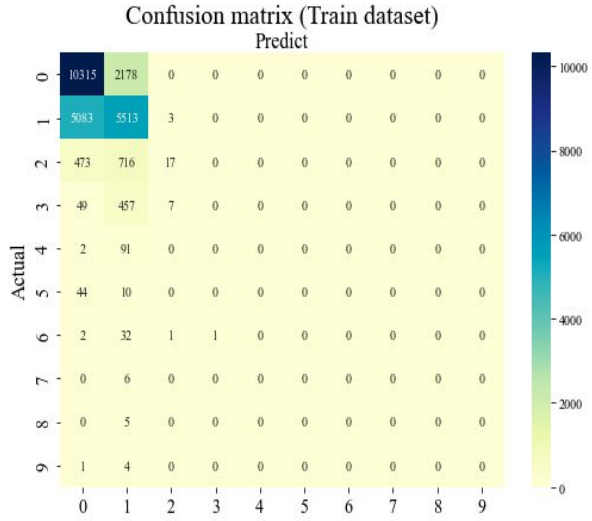


K-Nearest Neighbours

Results

- Although, confusion matrix illustrates that KNN model doesn't identify low probability poker hand
- k-NN is a distance based algorithm
- Closeness of two input features doesn't guarantee that they belong to similar classes





K-Nearest Neighbours

Results

- Although, confusion matrix illustrates that KNN model doesn't identify low probability poker hand
- k-NN is a distance based algorithm
- Closeness of two input features doesn't guarantee that they belong to similar classes

Sample	S1	C1	S2	C2	S3	C3	S4	C4	S5	C5	Class
a	4	1	4	2	4	3	4	4	4	5	8
b	4	2	4	3	4	4	4	5	4	6	8

Distance = 2.236

Sample	S1	C1	S2	C2	S3	C3	S4	C4	S5	C5	Class
a	4	1	4	2	4	3	4	4	4	5	8
c	4	1	4	2	4	3	4	4	3	6	0

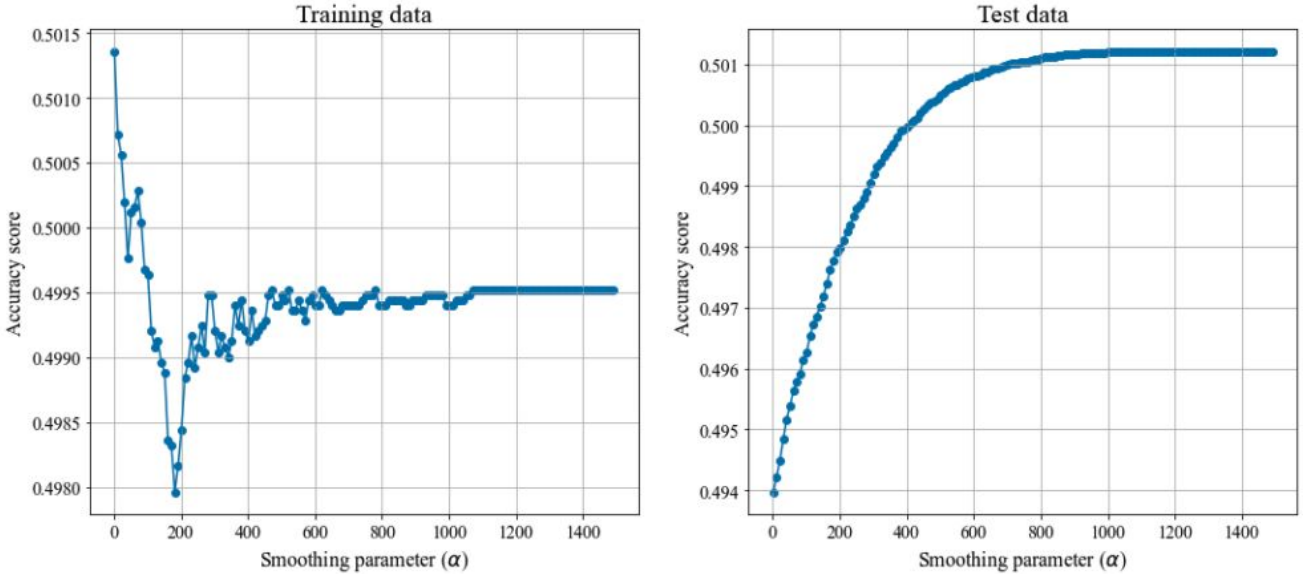
Distance = 1.414

Categorical Naive Bayes

Parameter Tunning

Search Space: {1, 11, 21, …, 1491}

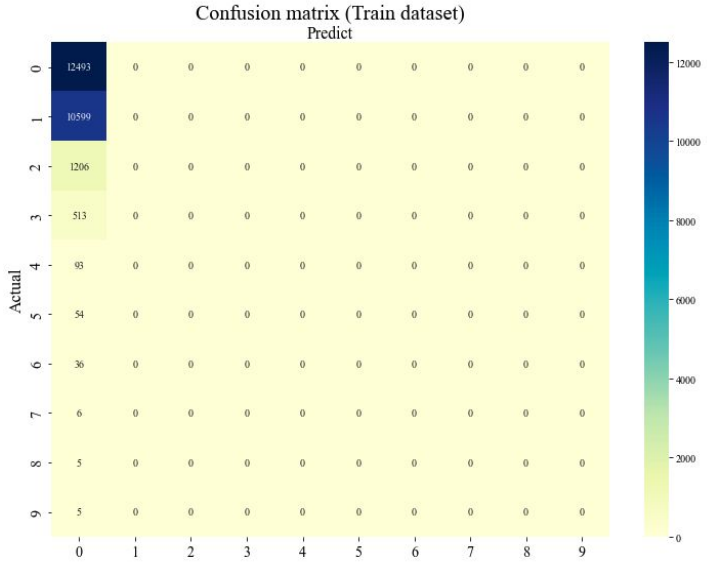
- Maximum accuracy on test dataset is obtain for alpha=1321.

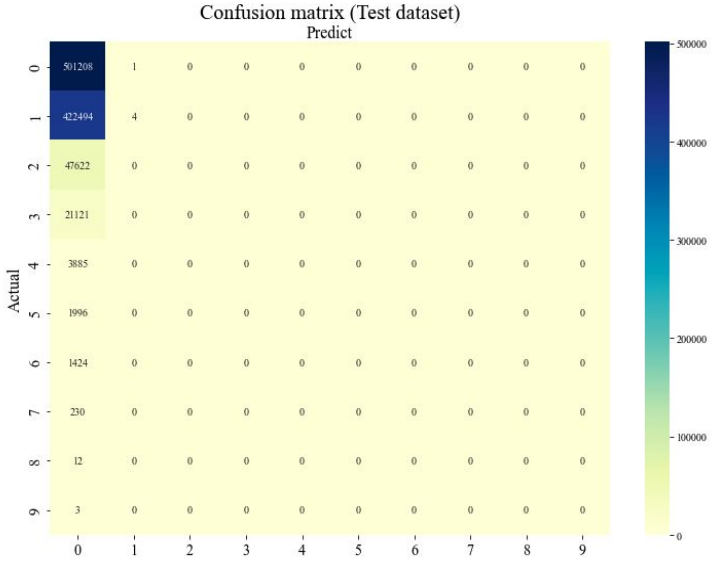


Categorical Naive Bayes

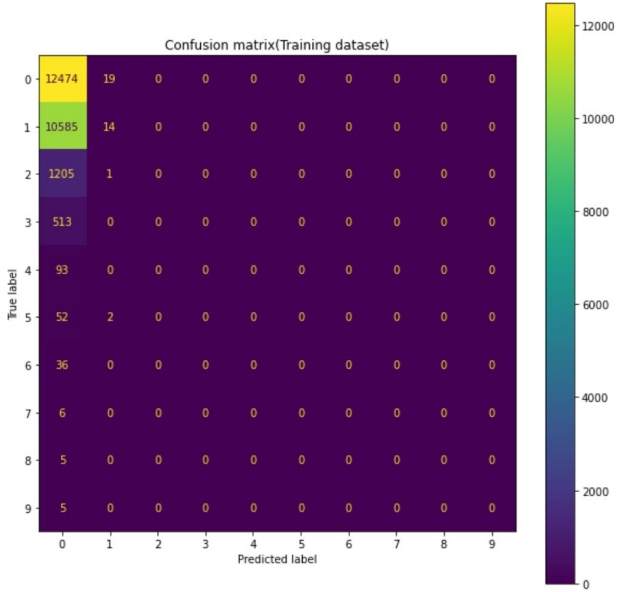
Results

- Here, confusion matrix shows that model doesn't identify poker hands accurately
- Feature distribution is baised to the frequency of class values





Linear Discriminant Analysis



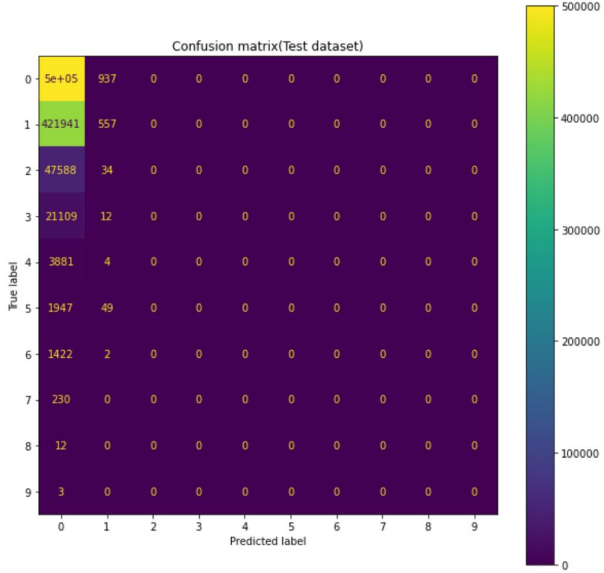
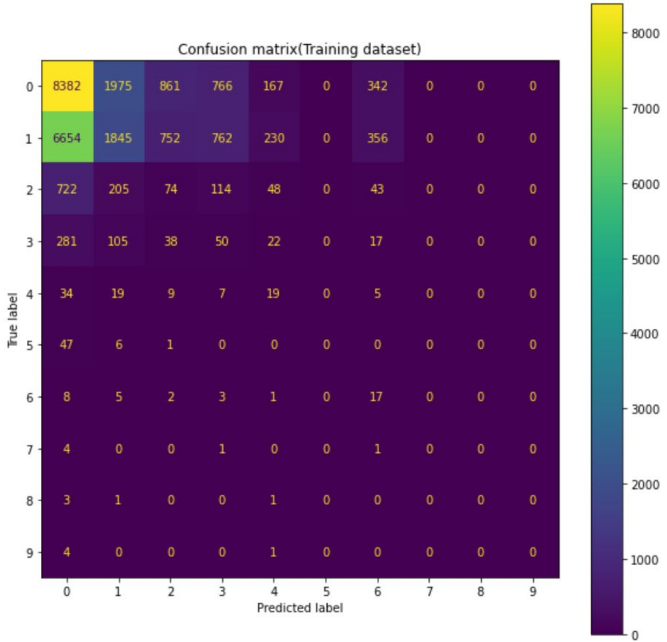


Table: LDA Training Result

Dataset	Training data	Test data
Accuracy	49.93%(12488/25010)	50.08%(500829/1000000)

Quadratic Discriminant Analysis



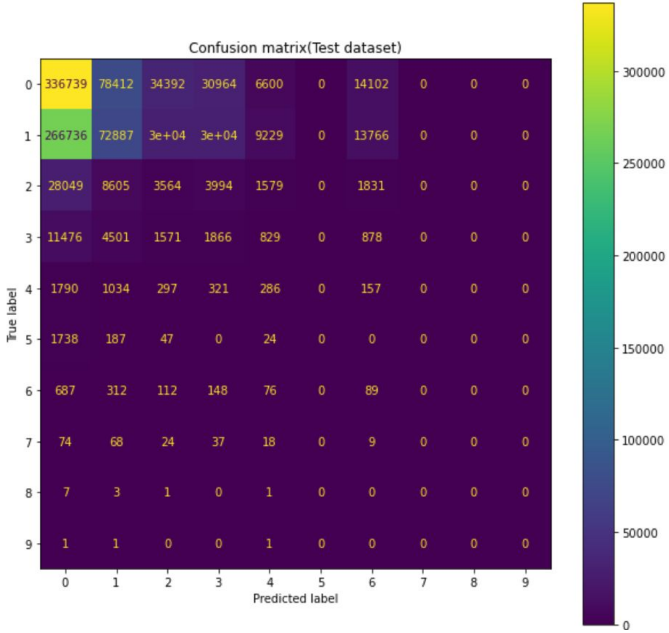
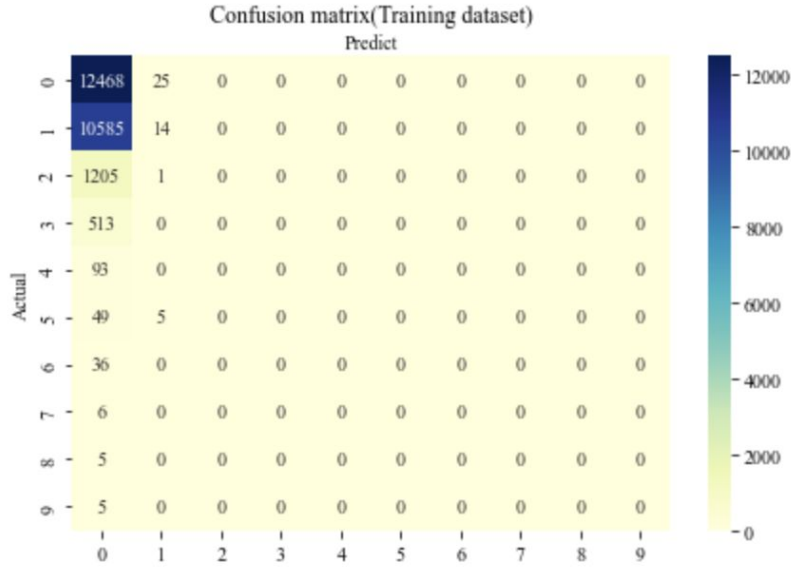


Table: QDA Training Result

Dataset	Training data	Test data
Accuracy	41.53%(10387/25010)	41.54%(415431/1000000)

Logistic Regression



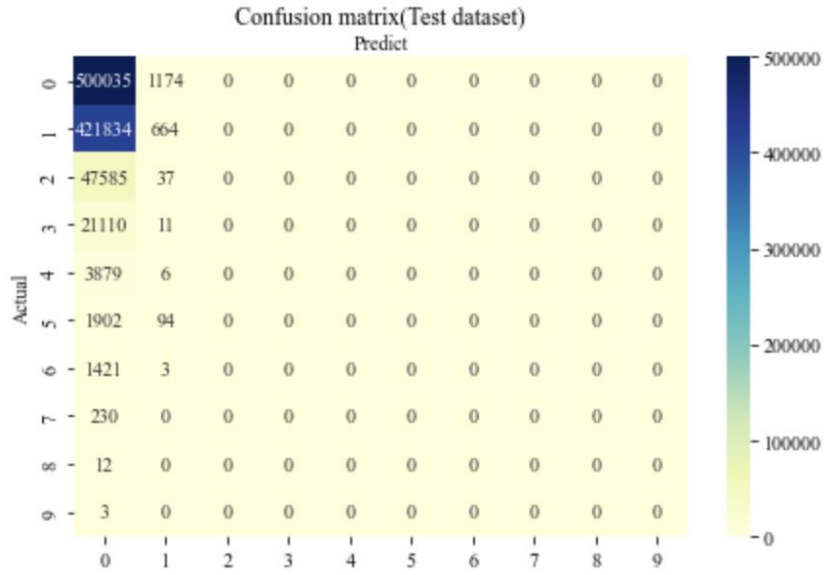
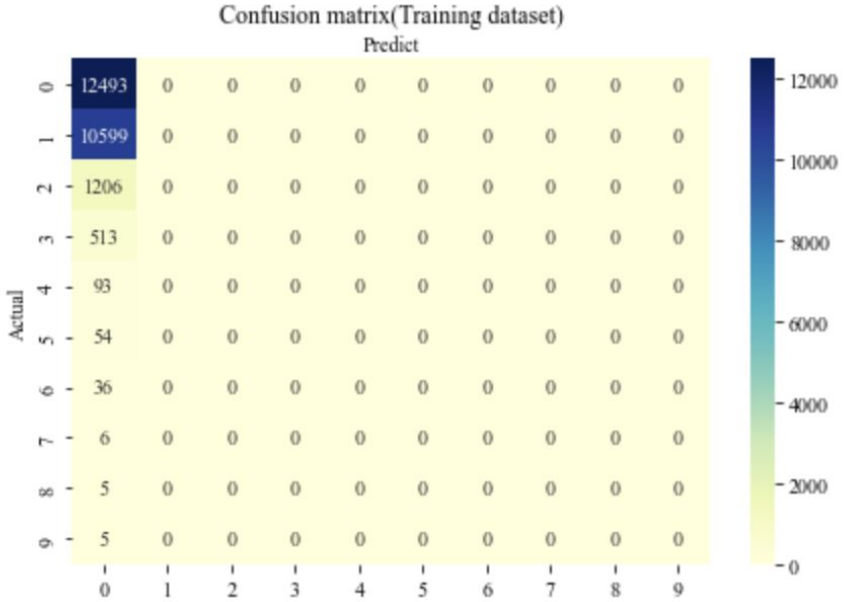


Table: Logistic Regression Result

Dataset	Training Data	Test Data
Accuracy	50.09%	49.93%

Support Vector Machines



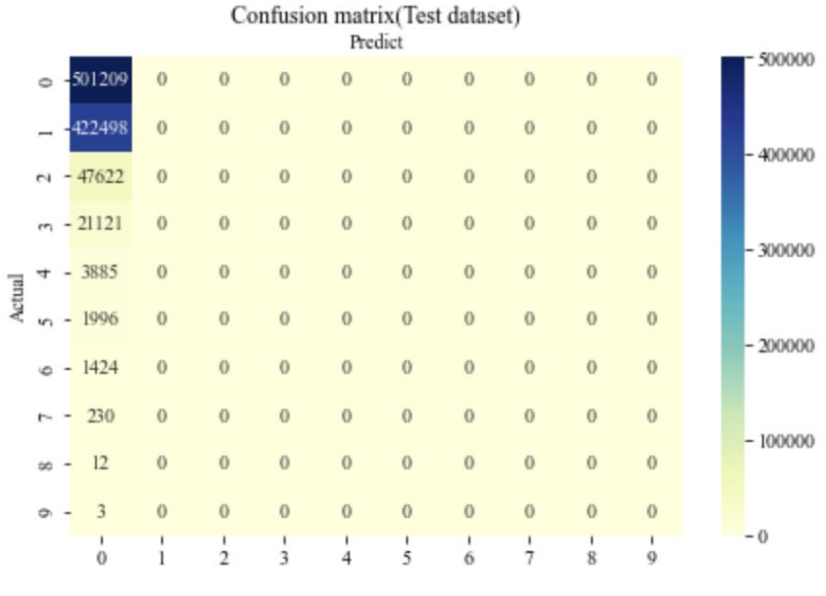
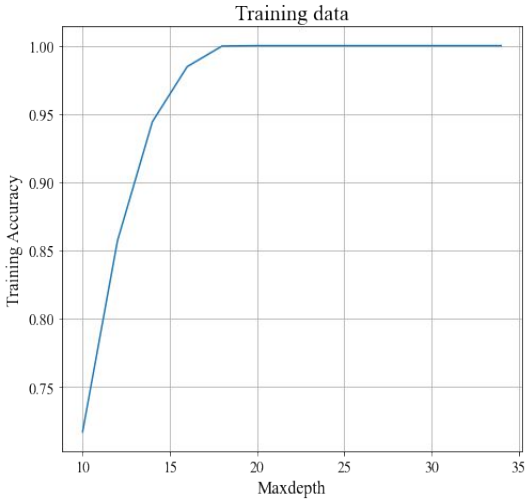
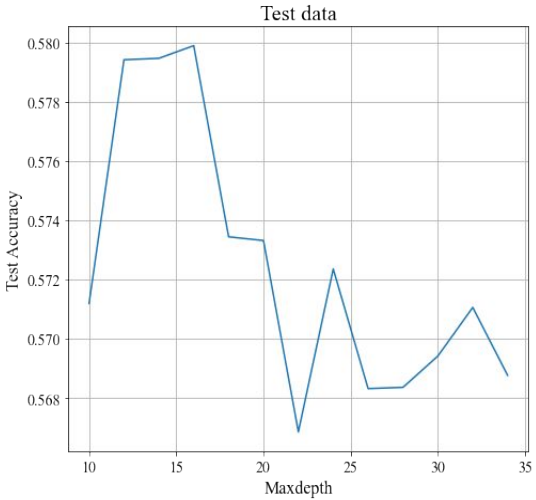


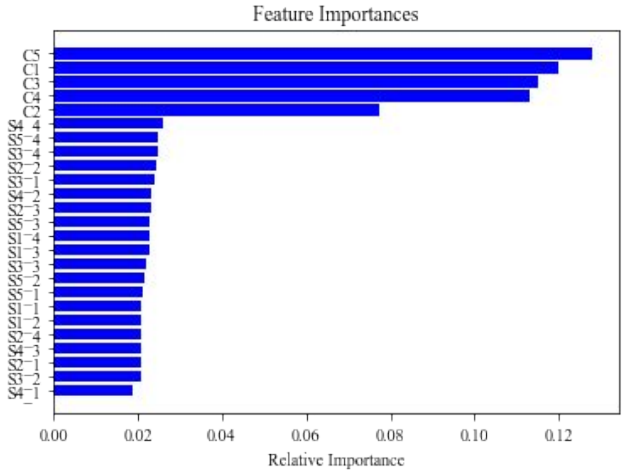
Table: SVM Result

Dataset	Training Data	Test Data
Accuracy	50.05%	49.88%

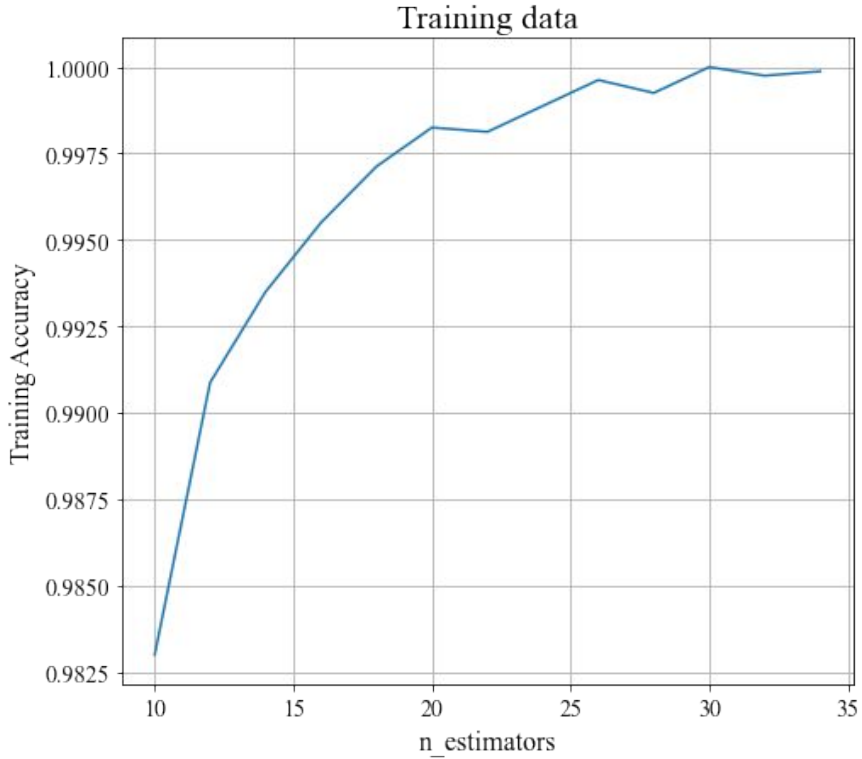
Decision Tree

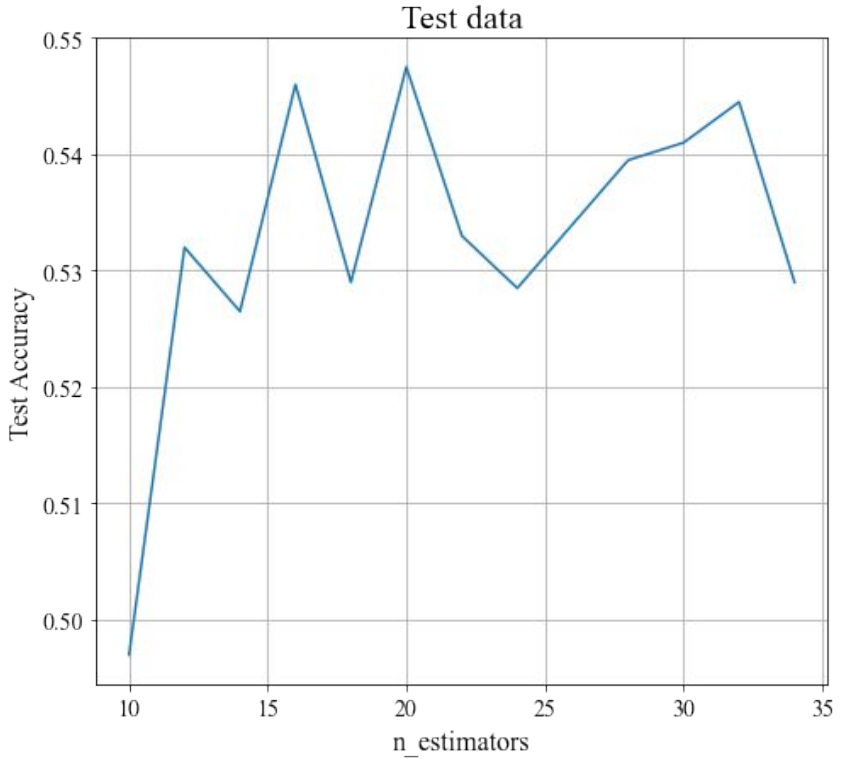




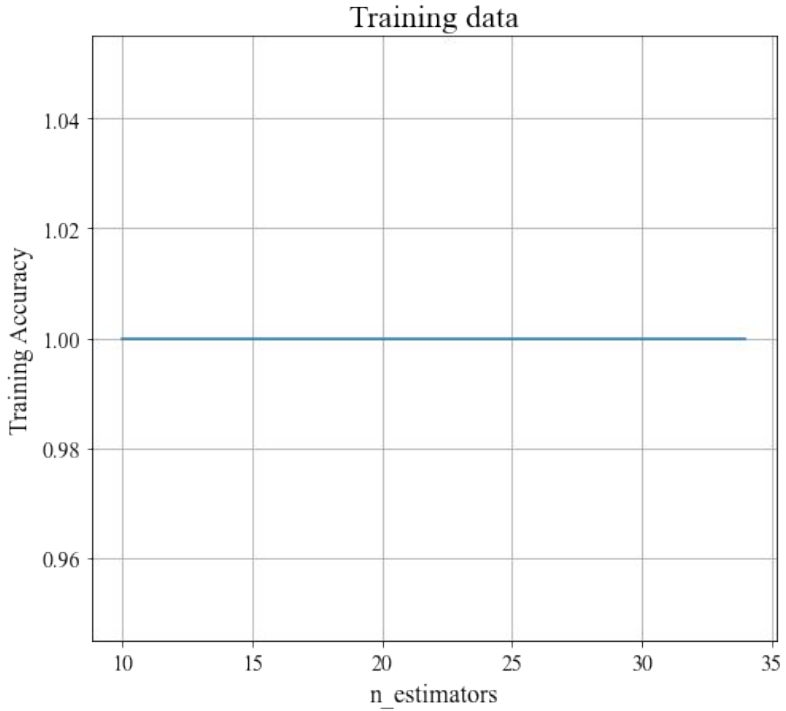


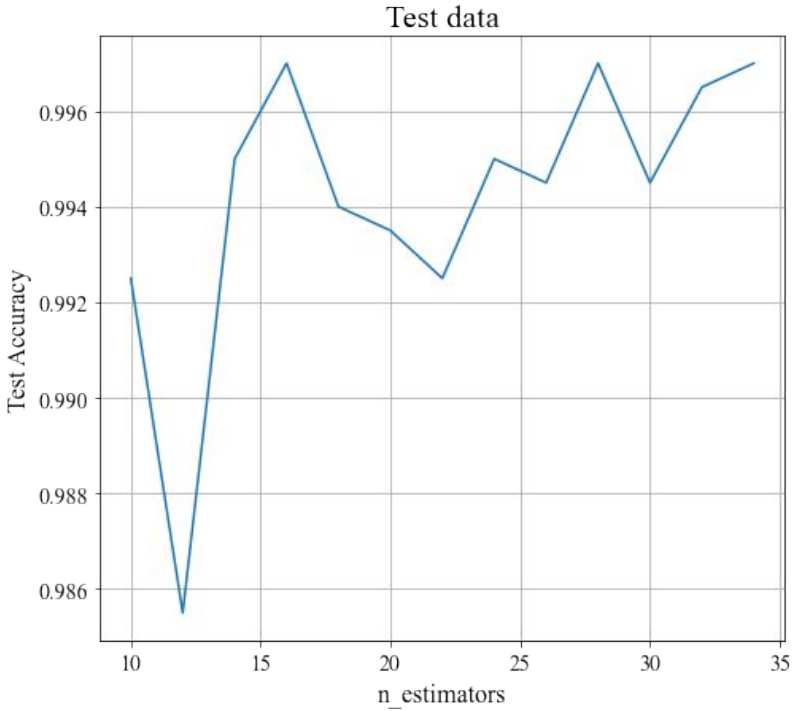
Random Forest - Using Binary Suit Data (25 sample features)





Random Forest - Using Numerical Suit Data (10 feature samples)





Neural Networks- first explorating using binary suit data (25 sample features)

Using nn.Linear(25,15)

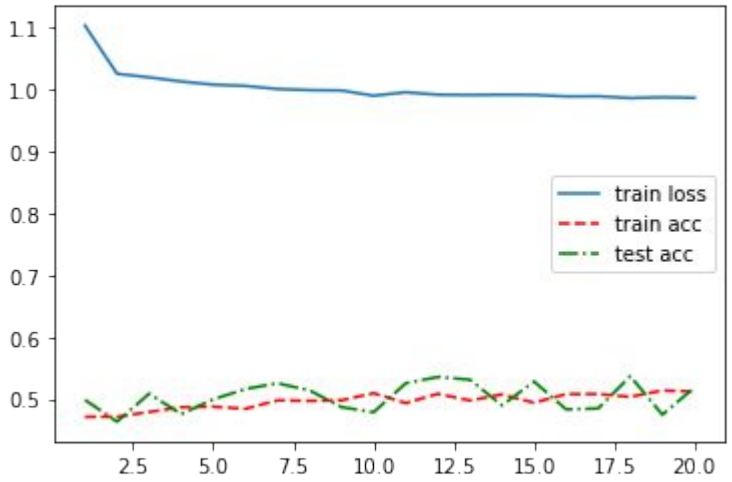
nn.ReLU()

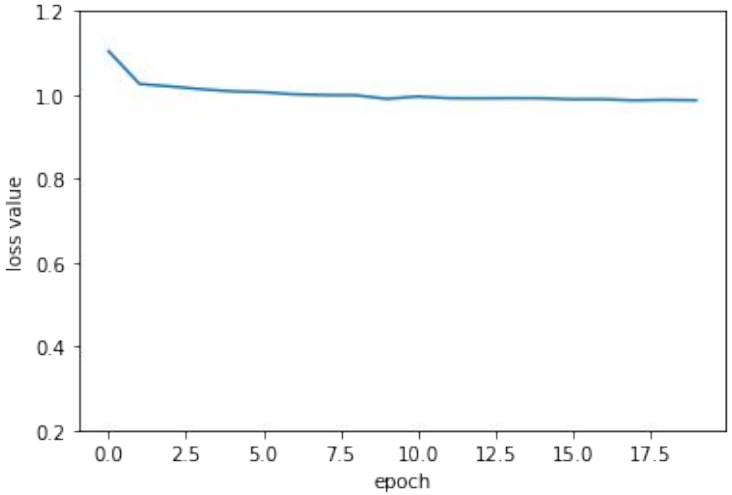
nn.Linear(15,10)

nn.ReLU() in sequence

```
num_epochs = 20
loss_vals, acc_train, acc_test = train_network(model,
```

```
Loss function value at epoch 5: 1.0062876597046853
Loss function value at epoch 10: 0.9960532633960247
Loss function value at epoch 15: 0.9893221917748451
```





Neural Networks- using binary suit data but adding more intermediate layers

Using nn.Linear(25,20),

nn.ReLU(),

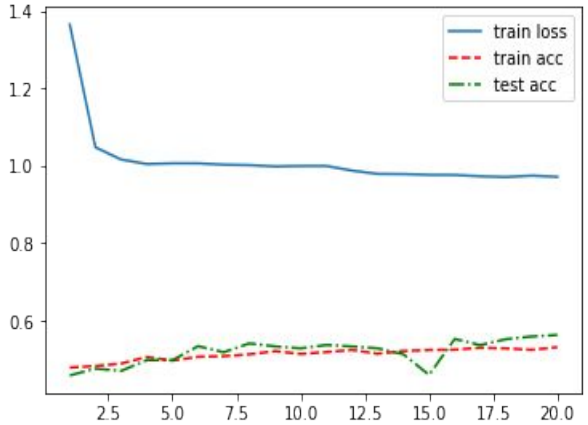
nn.Linear(20,15),

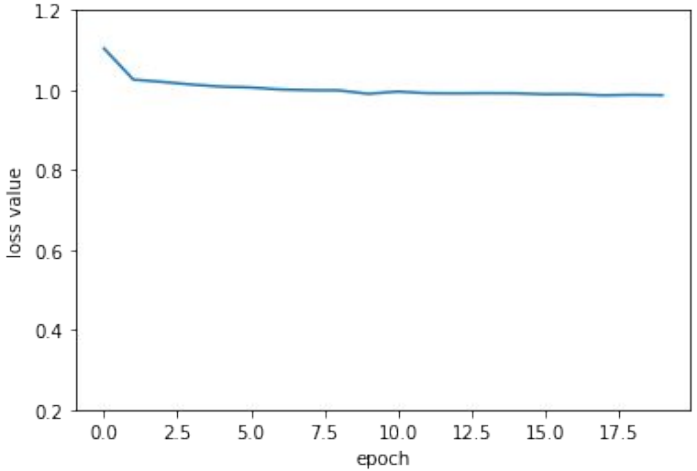
nn.Linear(15,10),

nn.ReLU()

```
num_epochs = 20
loss_vals, acc_train, acc_test = train_network(model,
```

```
Loss function value at epoch 5: 1.0054971262812615
Loss function value at epoch 10: 0.9987490327656269
Loss function value at epoch 15: 0.9755916541814804
```





Neural Networks- now using numerical suit data (10 sample features)

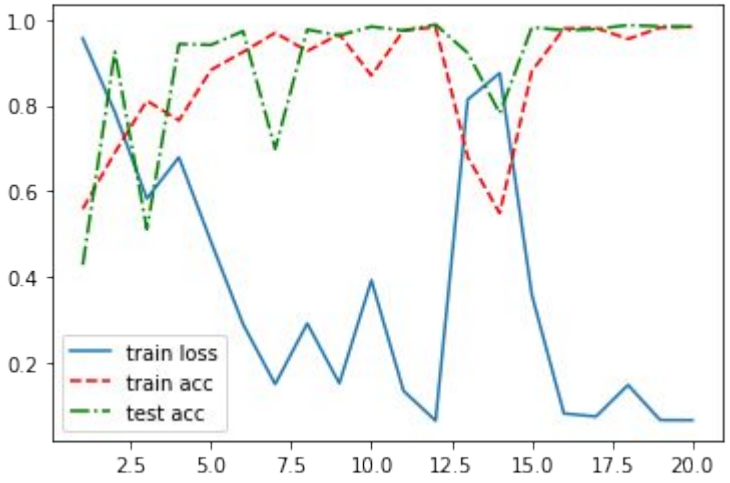
Using nn.Linear(10,20)

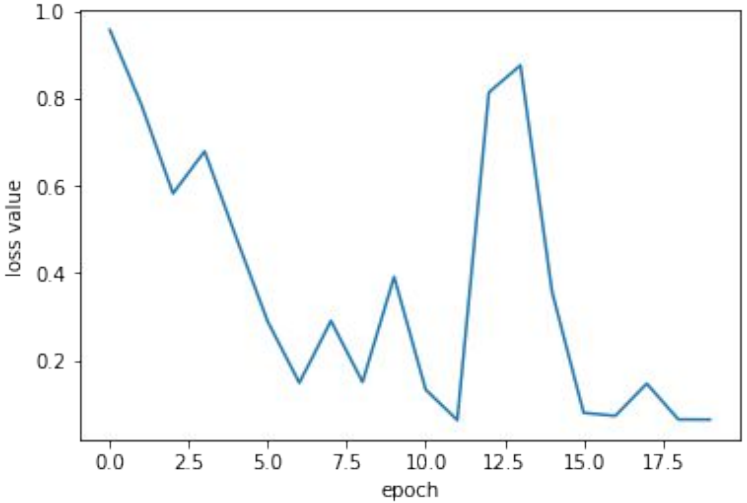
nn.ReLU()

nn.Linear(20,10)

```
num_epochs = 20
loss_vals, acc_train, acc_test = train_network(model,
```

```
Loss function value at epoch 5: 0.2887709668057505
Loss function value at epoch 10: 0.13329999336390755
Loss function value at epoch 15: 0.08048381049033196
```





Neural Networks- Using Raw data (10 features)

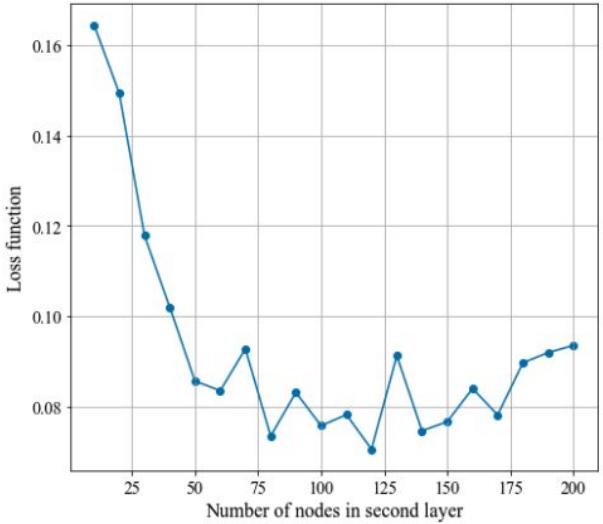
Python package: **keras**

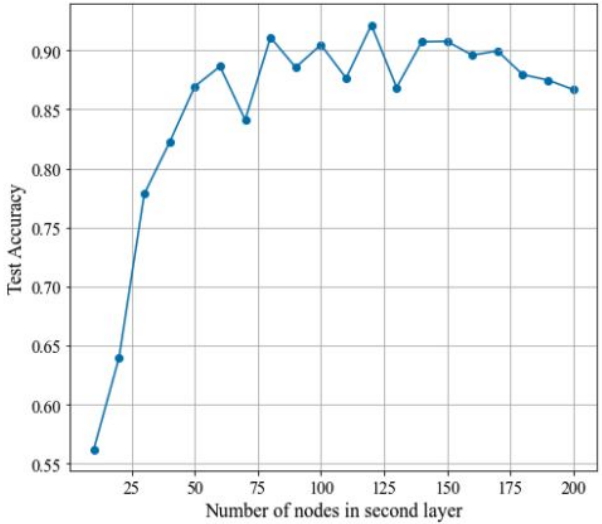
Three layers structure is used in this algorithm with number of nodes as [1.5*alpha, alpha, 10]

Search Space for alpha: {10, 20, …, 200}

Activation function: Rectified linear unit (Relu), Softmax function

Epoch = 250



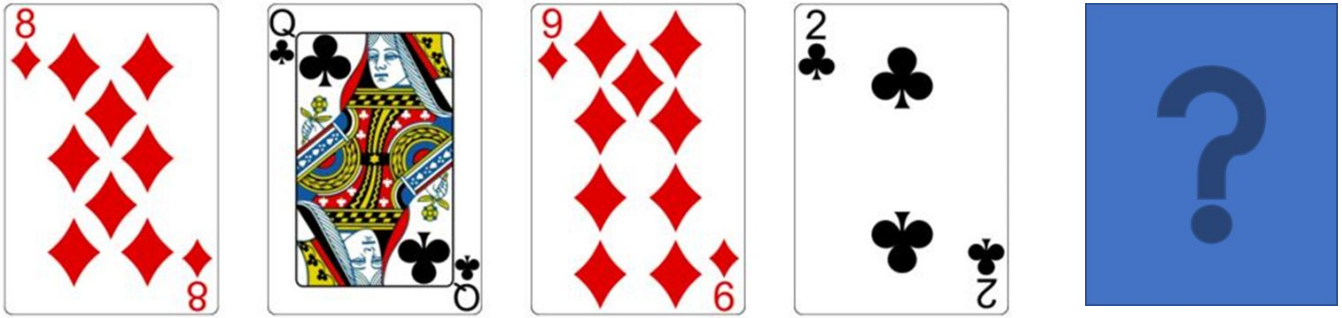


Our Conclusions

- Although poker hand recognition was initially considered by us a hard classification problem for a machine since it involves highly conceptual human language definitions that can correspond to numerous different card combinations, we proved that random forest and neural networks can achieve near perfect prediction accuracy
- After experimenting with 9 major classification methods with numerous parameter tuning on all of them, we feel confident that the decision boundary for this problem is linear
- Our major finding for this project is that: the process of artifically engineering features before feeding the data into the model can potentially dilute the information density contained in each feature and result in adverse effect on the model prediction accuracy.

  However, on the other hand, if we feed the model with fewer but information rich sample features and apply linear transformations(such as in the neural network algorithm) to them, even the intermediate linear transformations' outputs results in higher dimensions than the original input, the model is likely to better capture the essential nature of the data and thus produce better prediction results

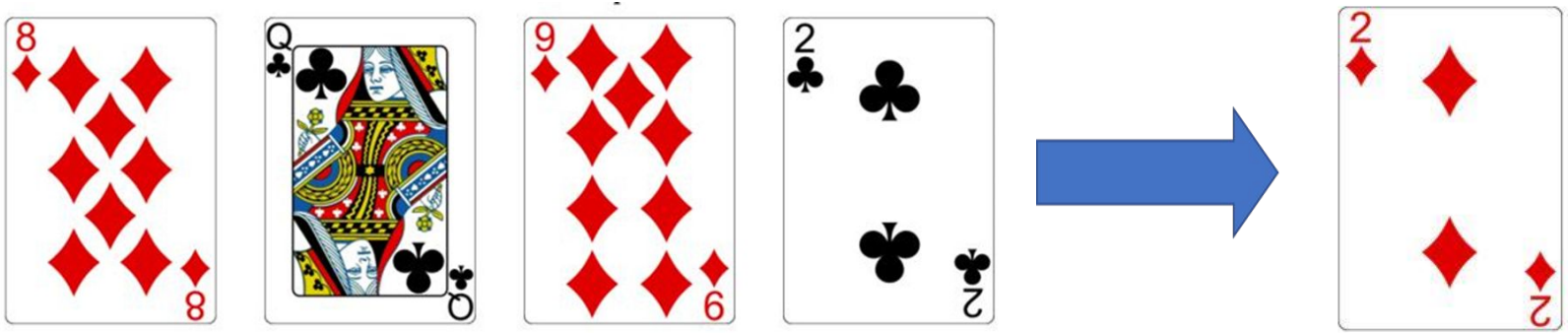
Future Extension: A Bayesian problem?



A Bayesian problem?

If the algorithm knows the hand category information, will it update its prediction on the fifth card in the right direction?

Ex: Given ‘one pair’



References

- http://archive.ics.uci.edu/ml/datasets/Poker+Hand
- https://www.apsl.net/blog/2017/07/18/using-linear-discriminant-analysis-lda-data-explore-step-step/
- https://virgoady7.medium.com/poker-hand-prediction-7a801e254acd