Each member in the group is expected to:

1. Explain dataset and problem.

2. Separately develop their chosen ML classifier (choose one from: Naïve Bayes, Random Forests and Support Vector Machine) and also a neural network classifier.

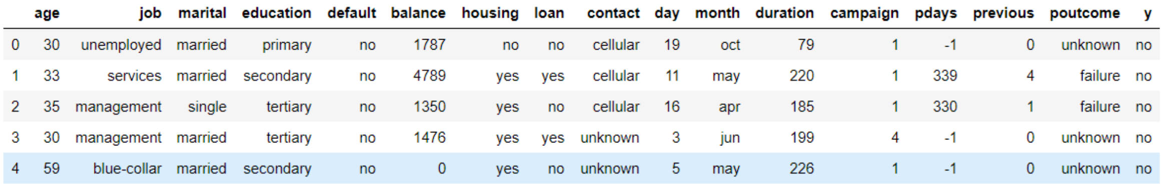
3. Explain how their chosen ML technique works.

4. Explain how network structure and hyperparameters were optimised for the neural network.

5. Analyse and compare accuracy and training time results for their chosen ML technique and neural networks.

This project is about applying two types of Machine Learning algorithms - a support vector classifier and a neural network classifier to solve a problem of Bank Marketing binary classification. The raw data come from UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing. The dataset describes different clients have different choices on whether to subscribe for a term deposit. As Table 1 illustrates, one client's choice is based on his or her features, which contain age, type of job, education level and so on. The models are established to differentiate customers of term deposit from others and prepare for future prediction.

Table 1 the first five rows of bank.csv



	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no

All variables have experienced some of the pre-processing techniques and feature engineering technique. As the scale of all numerical variables of attributes are relatively large, they are converted to be continuous and squeezed into the range of 0 to 1 through MinMaxScaler (Alice Zheng and Amanda Casari, 2018). All categorical variables of attributes are transformed through Label Encoder and one-hot encoding. The initial output variable is a string "Yes" or "No" and is also converted to integers 1 and 0 through Label Encoder. In the following steps, the data set is balanced and split to a training test (70% of the whole set) and a testing test (30% of the whole set) through train_test_split().

This program utilizes the Support Vector Classifier firstly. As SVC is sensitive to the scale of input features, it can run more quickly benefiting from the MinMaxScaler before (Alice Zheng and Amanda Casari, 2018). In terms of SVC's principles, it has the capability to produce decision boundaries generally. This is detailed below with classifying two-dimensional data. Starting from linear SVC, it has the ability to separate two classes of points with one straight line (Figure 1) (Jake VanderPlas, 2016). Nevertheless, those three lines may not enough to fit more data. For example, for a new data point marked by X

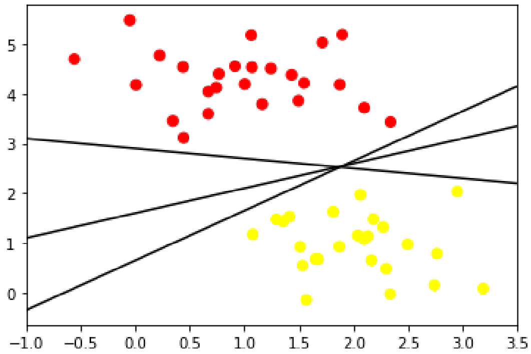


Figure 1

(Figure 2), different lines may produce different classification of this point. In other words, not all lines displayed here have great generalization ability. In order to prevent this, SVC is designed to draw straight lines with a margin of some width around as displayed in Figure 3 (Jake VanderPlas, 2016). For the best model fitting the training set, it has the maximal value of the margin and the best ability of generalization (Jake VanderPlas, 2016). A C parameter is utilized to represent the hardness of the margin. As shown in the Figure 4, it is evident that the margin is softer and more points can lie in it when C is smaller. For a big C such as 10 in the Figure 4, the margin is too hard for points to lie in it. However, as Figure 5 tells, the linear decision boundary cannot fit every kind of data. But principles of linear SVC are similar to those of SVC with “rbf” kernel, in which data are projected into higher-dimensional space and classified by a plane (Figure 6), and those of SVC with “poly” kernel, in which data are also not classified by linear decision boundary but by curves in two-dimension space (Figure 7) (Jake VanderPlas, 2016). All principles above can be expressed as Math formula and behave as theoretical support for the svm.SVC function.

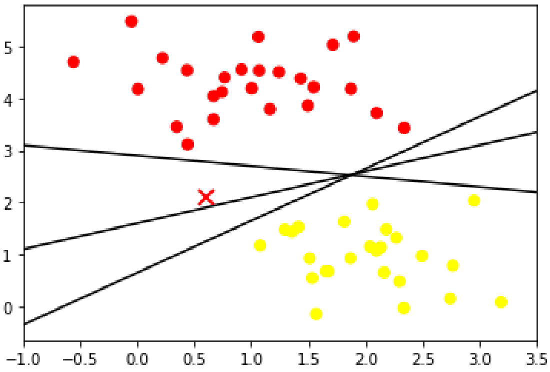


Figure 2

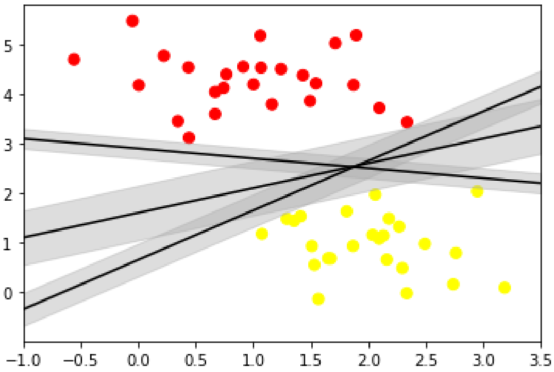


Figure 3

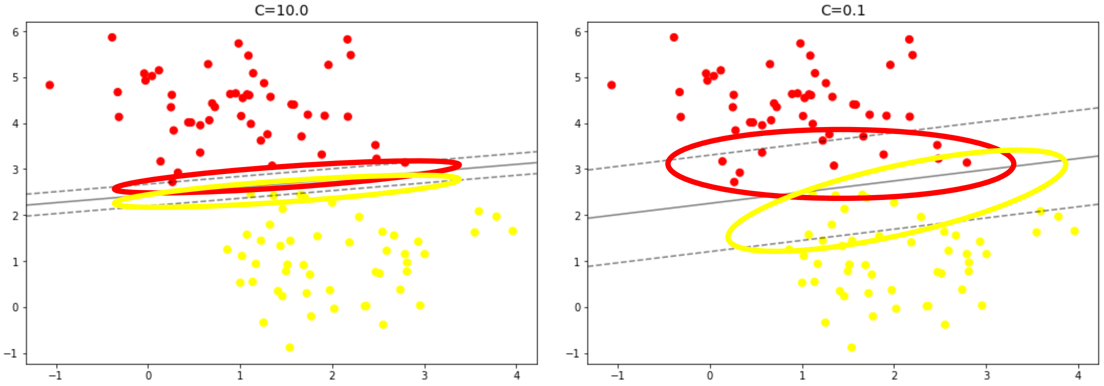


Figure 4

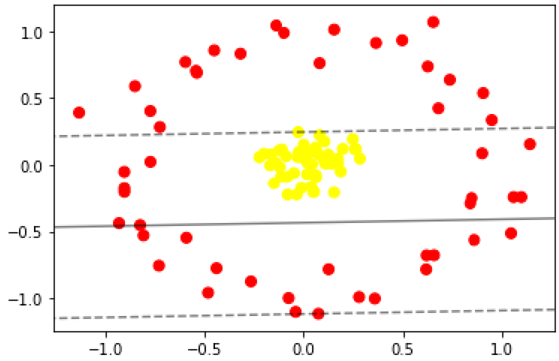


Figure 5

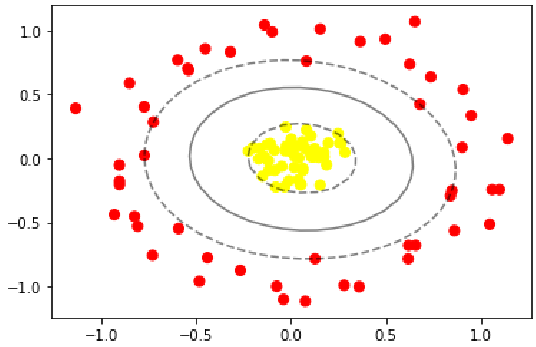


Figure 6

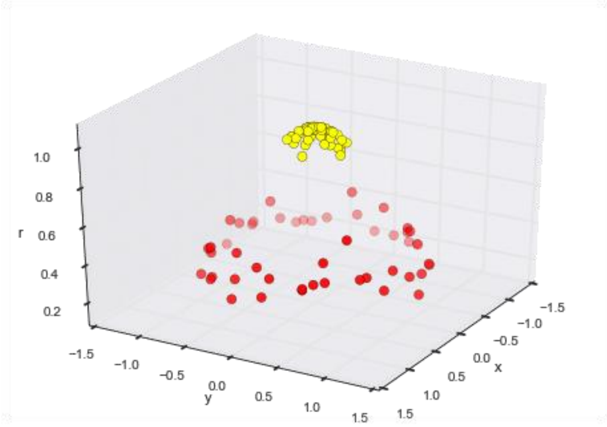


Figure 7

SVC is also adopted to classify multi-dimensional data, such as solving the problem of this project (Jake VanderPlas, 2016). Although it may be impossible to plot multi-dimensional data and dividing lines for them (especially for whose dimension is over 3), SVC is still able to working on making a classification of them (Jake VanderPlas, 2016). Hence, this program applies SVC with kernel “linear”, “rbf” and “poly” to fit the training set respectively. Firstly, in each kind of SVC, GridSearch is utilized to find the best parameter. The SVC model will be over-fitting when C is too big while the model will be under-fitting if C is too small (Andrew Ng, no date). According to the scikit-learn documentation (2019), it is recommended to choose C from 0.001 to 1000. As the grid is not relatively large, this program chooses 5 common C parameters for each kind of SVC model, most of which are distributed exponentially far apart from 0.005 to 10 (scikit-learn documentation, 2019). In addition, the “rbf” SVC has another important parameter – gamma, which specifies the radius of one support vector’s influence (scikit-learn documentation, 2019). Too big gamma causes over-fitting while too small gamma causes under-fitting (scikit-learn documentation, 2019). According to the scikit-learn documentation (2019), it is recommended to choose gamma from 0.001 to 1000. For the relatively small grid here, this program adopts 7 gamma parameters from 0.0001 to 5. From comparison, the linear SVC with C = 10 is the best model of all SVC models. Furthermore, from the learning curve of the best model of each kernel, none of them is under-fitting or over-fitting.

Then the program utilizes the model of Neural Network. When hyperparameters of Neural Network are selected, a balance of the model’s accuracy and efficiency needs to be ensured. Additionally, according to Andrew Ng (No date), learning curve is the most important hyperparameter. Then the number of neurons and the size of batch come after it (Andrew Ng, no date). This program chooses to focus on optimizing the number of neurons and the size of both batch and epoch.

Firstly, this model has 2 layers. As this program has 76 attributes, the neurons' number in the input layer is designed to be 38 or 76. In terms of current recommended activation function of the input layer, “ReLU” is a great choice. It can improve networks' performance by increasing non-linearity, prevent over-fitting as well as boost efficiency (Xavier Glorot, 2011). As the output y has only two discrete values of {0,1}, the output layer can have 2 neurons (Junming Zhang, 2018). The output layer's “Sigmoid” activation function fits this problem of binary classification very well. The range of the Sigmoid function (Figure 8) can scale the output value between 0 and 1 (Jason Brownlee, 2019). Correspondingly, for the binary classification, the loss function can be “binary_crossentropy”. In addition, “adam” is a good default choice because of its high efficiency and good performance (Sebastian Ruder, 2017; Fabio M. Graetz,2018). And the metrics of this model should be “accuracy” as the metrics of this project is “accuracy”.

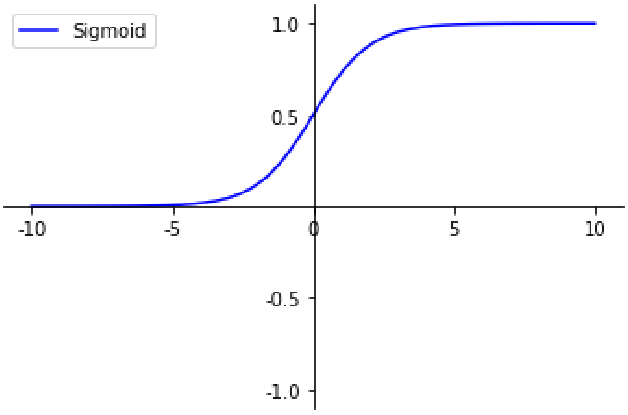
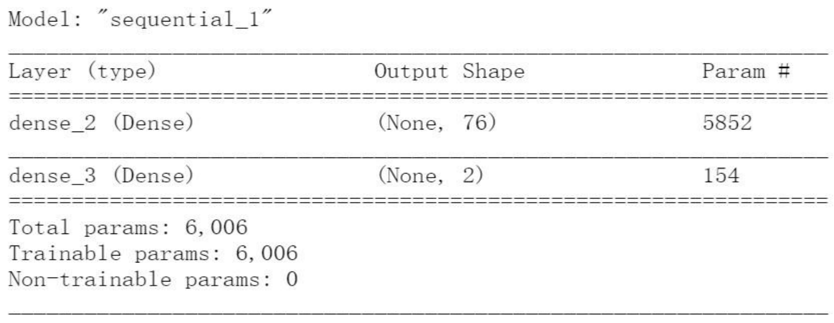


Figure 8

Secondly, it needs to take care when parameters are chosen in GridSearch. “One epoch” refers to training once with all samples in the training set while “batch size” refers to the size of samples which are taken from the training set for one iteration. The size of epoch and batch size should be reasonable. As GridSearch iterates over all parameters, if the parameters are too large this will be an extremely time-consuming task. Moreover, the computer may not have sufficient RAM for GridSearch in this case. For example, if “epoch = 1000” exists in the parameter combinations, this may cause memory overflow. Moreover, for a normal data set, if the batch size is too small, training data will be very difficult to converge, leading to under-fitting. And as the number of epochs increases, the model changes from under-fitting to over-fitting (SAGAR SHARMA, 2017; No name, Gradient Descent: All you need to know, 2018). As many models of common parameters have high accuracy but low generalization ability here, this project has tried to adjust hyperparameters multiple times to prevent over-fitting although the process has not been documented entirely. As a result, the best parameter of all trials without over-fitting or under-fitting is {'batch_size': 583, 'epochs': 10, 'neurons': 76} (Figure 9).



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 76)                5852
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 154
=================================================================
Total params: 6,006
Trainable params: 6,006
Non-trainable params: 0
_________________________________________________________________
```

Figure 9 The structure of the best model of all trials

As Figure 10 and Table 2 shows, when Neural Network is adopted to predict the testing set, its accuracy is roughly 62%, which is smaller than each

SVC model. As neural network has so many hyperparameters to adjust and the size of training set is not very large, the best model here may be not the best one in action.

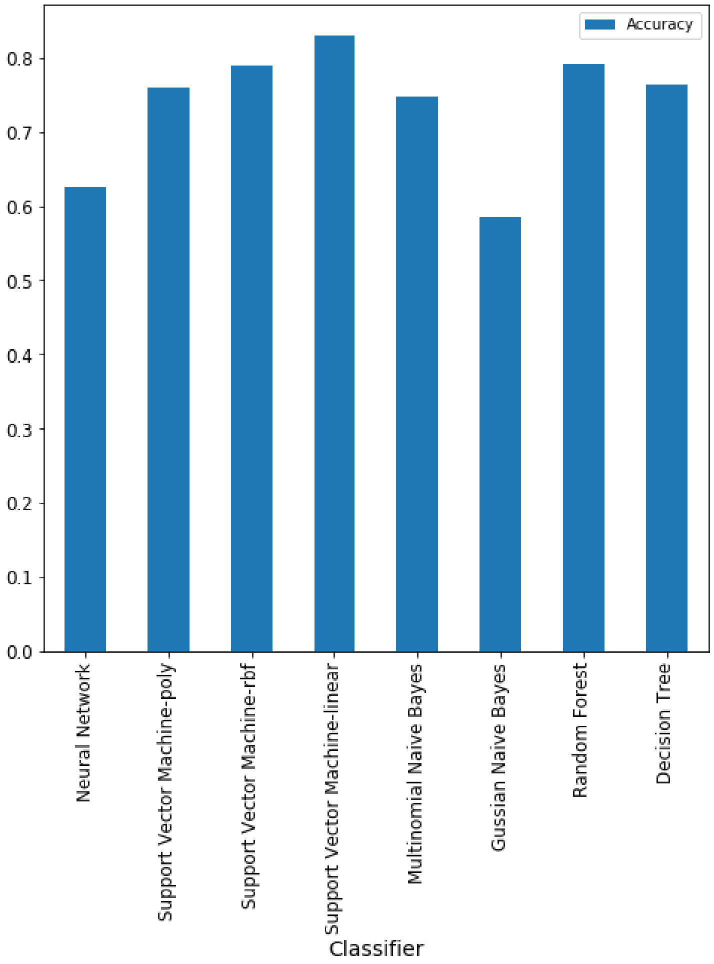
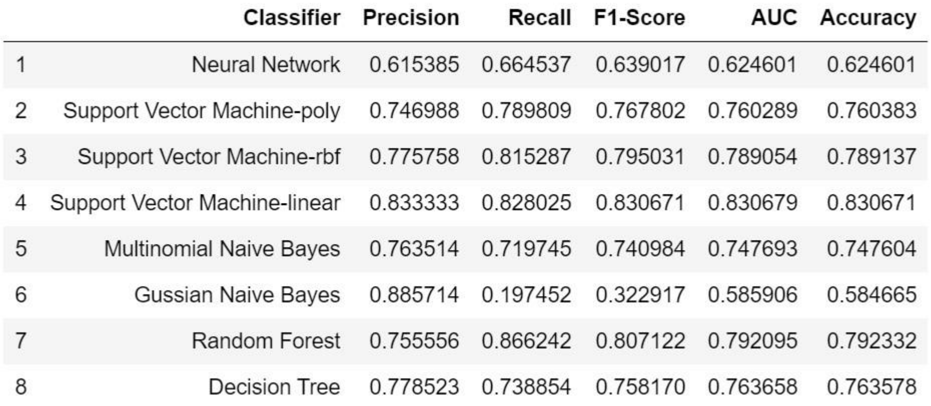


Figure 10

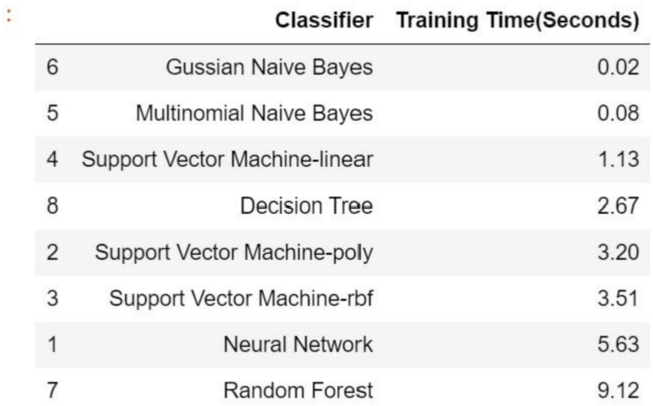
Table 2



	Classifier	Precision	Recall	F1-Score	AUC	Accuracy
1	Neural Network	0.615385	0.664537	0.639017	0.624601	0.624601
2	Support Vector Machine-poly	0.746988	0.789809	0.767802	0.760289	0.760383
3	Support Vector Machine-rbf	0.775758	0.815287	0.795031	0.789054	0.789137
4	Support Vector Machine-linear	0.833333	0.828025	0.830671	0.830679	0.830671
5	Multinomial Naive Bayes	0.763514	0.719745	0.740984	0.747693	0.747604
6	Gussian Naive Bayes	0.885714	0.197452	0.322917	0.585906	0.584665
7	Random Forest	0.755556	0.866242	0.807122	0.792095	0.792332
8	Decision Tree	0.778523	0.738854	0.758170	0.763658	0.763578

This program applies the time() function to calculate running time. Running time of all SVC models and the Neural Network model is big enough for comparison, so the timeit() function is unnecessary for them， which Gussian Naïve Bayes and Multinomial Naïve Bayes may need. Neural Network runs more slowly than each SVC model (Table 3). Moreover, time of adjusting hypermeters multiple times has not been added to running time. If this is added, Neural Network costs much more time than each SVC model.

Table 3



	Classifier	Training Time(Seconds)
6	Gussian Naive Bayes	0.02
5	Multinomial Naive Bayes	0.08
4	Support Vector Machine-linear	1.13
8	Decision Tree	2.67
2	Support Vector Machine-poly	3.20
3	Support Vector Machine-rbf	3.51
1	Neural Network	5.63
7	Random Forest	9.12

Reference List

Alice Zheng; Amanda Casari (2018) Feature Engineering for Machine Learning.

Fabio M. Graetz (2018) Why AdamW matters. Available at: https://towardsdatascience.com/why-adamw-matters-736223f31b5d

Jake VanderPlas (2016) Python Data Science Handbook.

Jason Brownlee (2019) Binary Classification Tutorial with the Keras Deep Learning Library. Available at: https://machinelearningmastery.com/binary-classification-tutorial-with-the-keras-deep-learning-library/

Junming Zhang (2018) Analysis of Neural Network on Bank Marketing Data. users.cecs.anu.edu.au › ABCs2018.

No name (2018) Gradient Descent: All you need to know. Available at: https://mc.ai/gradient-descent-all-you-need-to-know/

SAGAR SHARMA (2017) Epoch vs Batch Size vs Iterations. Available at: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

Scikit-learn documentation (2019) 1.4. Support Vector Machines. Available at: https://scikit-learn.org/stable/modules/svm.html#svm-classification

Sebastian Ruder(2017) An overview of gradient descent optimization. algorithms. arXiv.org, Jun 15, 2017.

Xavier Glorot, Antoine Bordes, Yoshua Bengio (2011) Deep Sparse Rectifier Neural Networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, PMLR 15:315-323.