STUDENT NAME: ONYEOGULU TOCHUKWU ROWLAND

STUDENT ID: 19175136

DALT7012: ADVANCED MACHINE LEARNING

COURSEWORK: SEMESTER 2, 2021-2022

WORD COUNT: TWO THOUSAND AND THIRTY WORDS

1.1. Introduction

Machine learning (ML) and Deep learning (DL) is a branch of artificial intelligence that deals with extracting knowledge from data [1]. It is a research field that combines mathematics, statistics and computer science. Over time, the application of ML and DL has become ubiquitous in our everyday life. Ranging from movie recommendations on our favourite TV platform to what product to buy from an online store and recognizing friends and family in our mobile photo application. Modern technology companies like Amazon, Facebook and Netflix have multiple ML and DL algorithms at their core. ML and DL is categorised into three domains which are supervised learning, unsupervised learning and reinforcement learning [2].

Performance Metrics plays a significant role in achieving the optimal ML or DL classifier. Therefore, selecting the most suitable performance metrics is an important aspect in discriminating and obtaining the optimal classifier. Although there is no consensus on which performance metric is better [3], the performance metric used for this project are Accuracy score, Area Under the Receiver Operating Curve and F-1 score.

2.0 Data Preparation and Exploration

The affirmative and conditional facial expression dataset was properly read into the python kernel in the Google Colab platform. The four dataset (a_affirmative, b_affirmative, a_conditional and b_conditional) were explored to check for missing data which confirms the presence of no missing data (see figure 2.1 below). In order to check for class imbalance, we plot the target variable for each dataset to ascertain if the distribution of examples across the two classes 0 (negative class) and 1 (positive class) is biassed or skewed. Figure 2.2, 2.3, 2.4 and 2.5 showed the distribution of examples among the two classes. From the graphs below, we can generalise that there exists a severe class imbalance for a_affirmative, a_conditional and b_conditional datasets which may affect our model performance. Hence, the b_affirmative dataset also suggests a class imbalance which is not severe. The pre-processing technique (scaling) was performed on all four dataset before implementing Support Vector Classification (SVC) and Artificial Neural Network (ANN) algorithm for classification. Scaling is an essential technique in ML and DL as it helps improve the performance of our classifier.

Figure 2.1: Python code to check for missing data

```
# checking for missing values in a_affirmative dataset
a_affirmative.isnull().sum().sum()
```

```
0
```

```
[39]
# checking for missing values in b_affirmative dataset
b_affirmative.isnull().sum().sum()
```

```
0
```

```
# checking for missing values in a_conditional dataset
a_conditional.isnull().sum().sum()
```

```
0
```

```
[41]
# checking for missing values in b_conditional dataset
b_conditional.isnull().sum().sum()
```

```
0
```

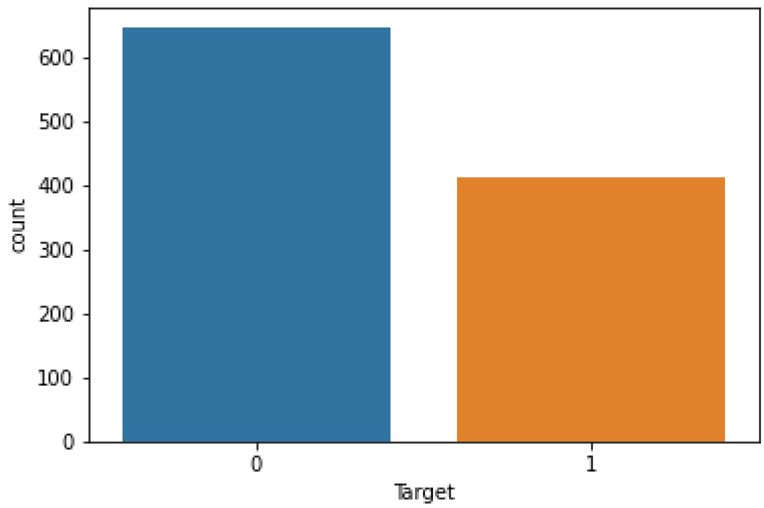


Figure 2.2: Checking for class imbalance in the a_affirmative dataset

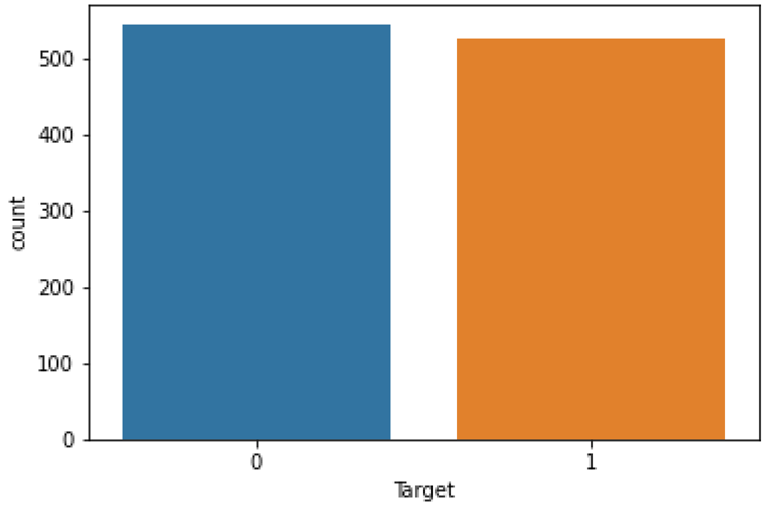


Figure 2.3: Checking for class imbalance in the b_affirmative dataset

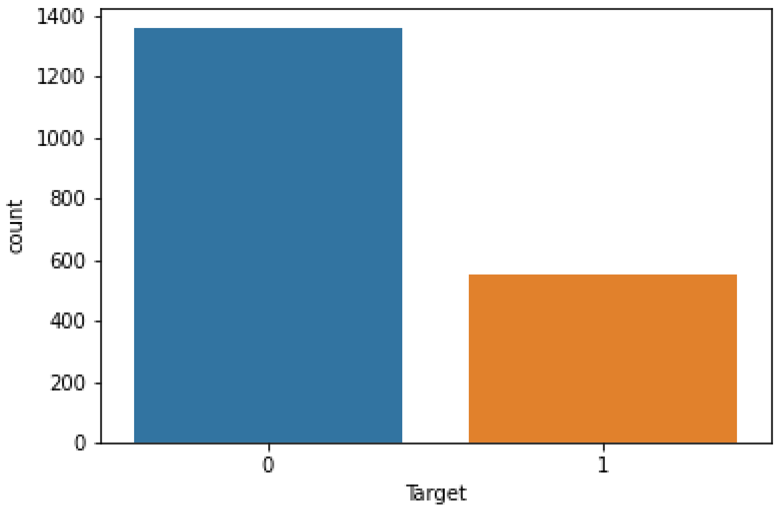


Figure 2.4: Checking for class imbalance in the a_conditional dataset

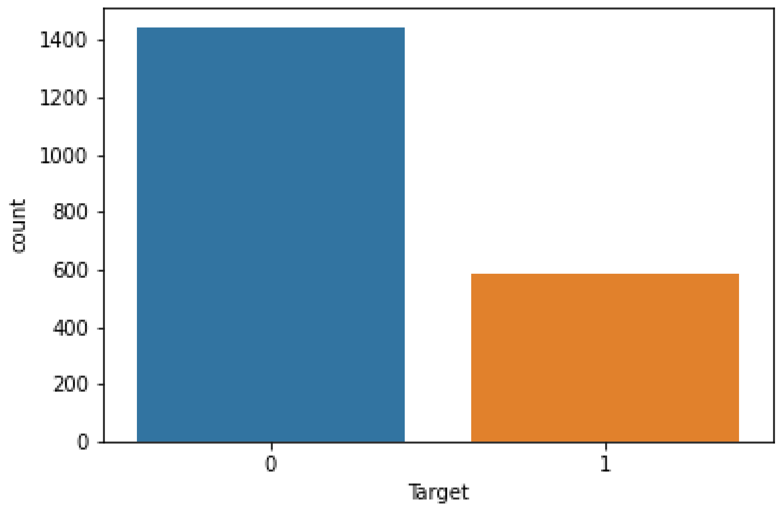


Figure 2.5: Checking for class imbalance in the b_conditional dataset

3.0 Support Vector Classification Model

Support Vector Machine (SVM) is a supervised ML algorithm first proposed by Vapnik which is used to solve both classification and regression problems [4]. This algorithm has since attracted a high level of interest across the ML research community for its capability of delivering higher model performance (with limited data) than other ML algorithms. However, the performance of SVM is sensitive to how its hyperparameters (C, kernel and gamma) are set. In order to obtain the best model parameter setting, there is a need for the user to conduct an extensive search through cross validation. This process called model selection is also applied in this coursework. In this work, two types of SVM algorithm were implemented;

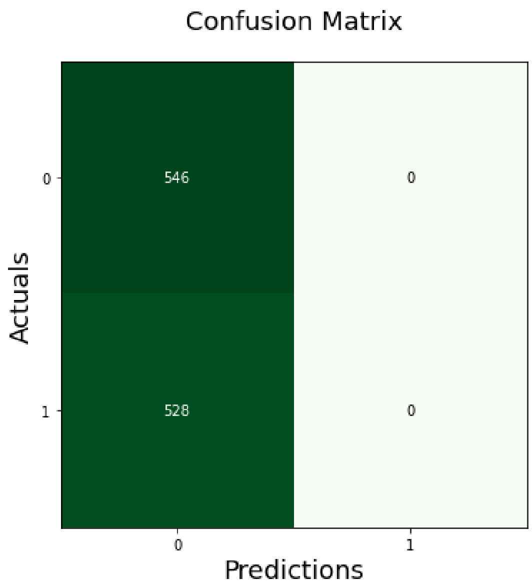
1. Soft linear SVM classifier with C parameter equal to 1 and linear kernel.
2. SVM classifier with five cross validation implemented with GridSearchCV for hyperparameter tuning for model selection.

3.1.0 Soft Linear SVM Classifier

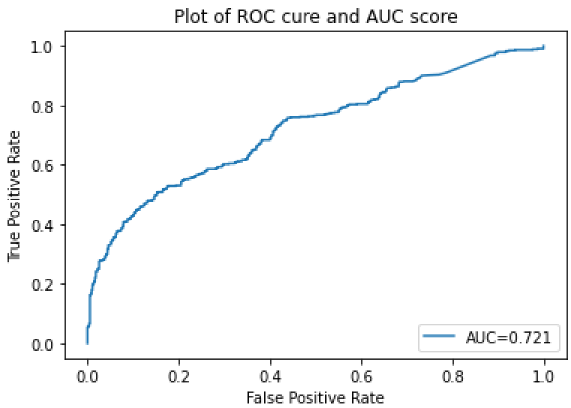
Python code below implements the soft linear SVM classifier without gridsearch cv and hyperparameter tuning. The SVM model with parameters (C=1, kernel=rbf) was used in training the different dataset considered for this coursework.

```python
from sklearn.pipeline import make_pipeline

def model(train_x,train_y,test_x,test_y):
    clf = make_pipeline(StandardScaler(),SVC(probability=True))
    # train model on training set
    clf.fit(train_x,train_y)

    # get modl parameters
    print("Model parametrs is:",clf.get_params(['svc__C']))
    # model prediction
    y_pred = clf.predict(test_x)
    # plot confusion matrix for the model prediction
    fig, ax = plot_confusion_matrix(conf_mat=confusion_matrix(test_y,y_pred), figsize=(6, 6), cmap=plt.cm.Greens)
    plt.xlabel('Predictions', fontsize=18)
    plt.ylabel('Actuals', fontsize=18)
    plt.title('Confusion Matrix', fontsize=18)
    plt.show()
    print(classification_report(test_y, y_pred))
    # creating ROC and AUC for the model
    #define metrics
    y_pred_proba = clf.predict_proba(test_x)[::,1]
    fpr, tpr, _ = metrics.roc_curve(test_y,  y_pred_proba)
    auc = metrics.roc_auc_score(test_y, y_pred_proba)

    #create ROC curve
    plt.plot(fpr,tpr,label="AUC="+str(round(auc,3)))
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.legend(loc=4)
    plt.title("Plot of ROC cure and AUC score")
    plt.show()
```

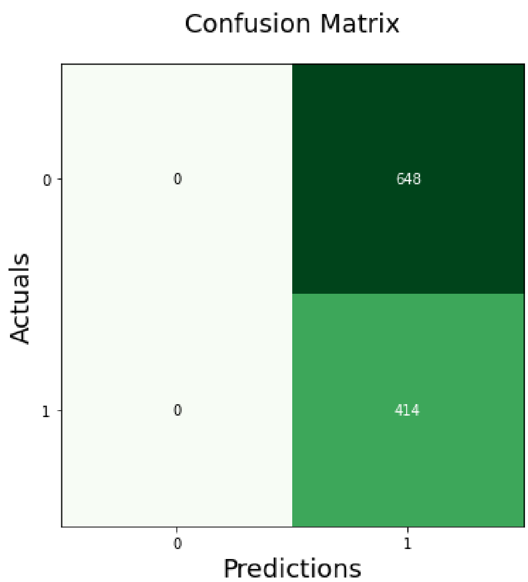
3.1.1 Model summary for a_affirmative as training set and b_affirmative as testing set



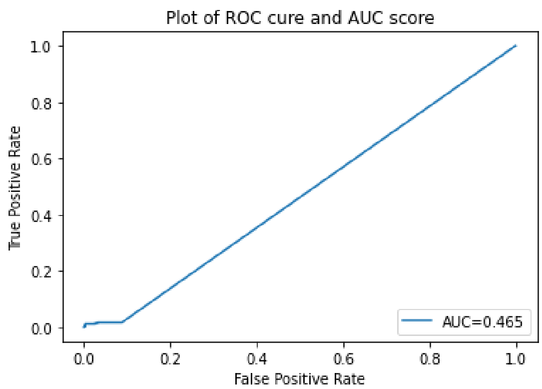
	precision	recall	f1-score	support
0	0.51	1.00	0.67	546
1	0.00	0.00	0.00	528
accuracy			0.51	1074
macro avg	0.25	0.50	0.34	1074
weighted avg	0.26	0.51	0.34	1074



3.1.2 Model summary for b_affirmative as training set and a_affirmative as testing set



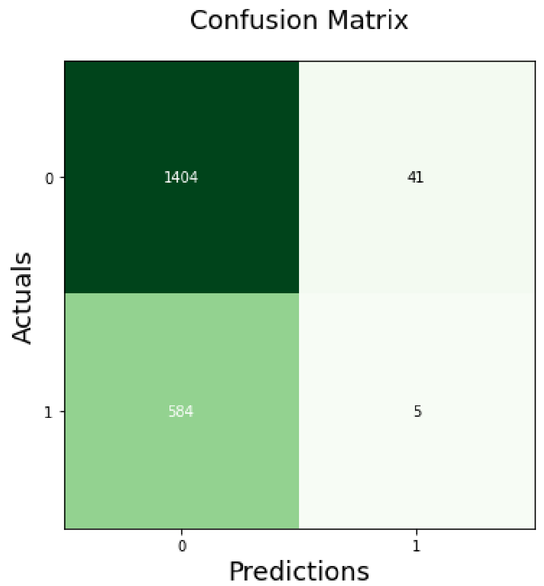
	precision	recall	f1-score	support
0	0.00	0.00	0.00	648
1	0.39	1.00	0.56	414
accuracy			0.39	1062
macro avg	0.19	0.50	0.28	1062
weighted avg	0.15	0.39	0.22	1062

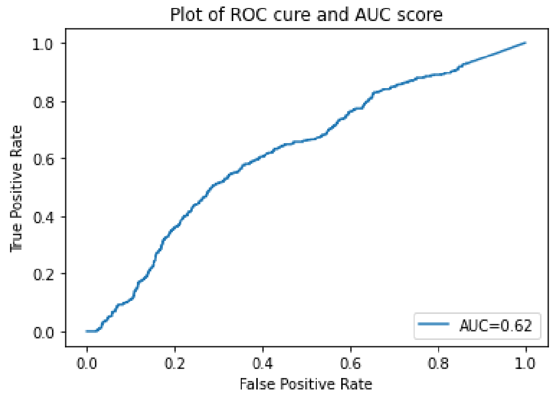


The figures in section 3.1.1 and 3.1.2 above provide the svc model summary for a_affirmative and b_affirmative. The confusion matrix shows that both models predicted only one class. This is caused by the class imbalance inherent in the dataset. The model summary shows that when a_affirmative was used as the training set, the model achieved a better model performance in all three performance metrics (accuracy:51%, auc:0.712, f1-score:0.67) than when b_affirmative is used as the train set.

3.1.3 Model summary for a_conditional as training set and b_conditional as testing set

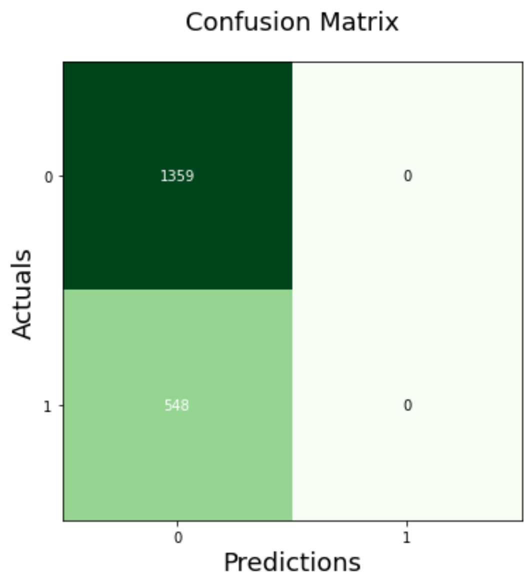
	precision	recall	f1-score	support
0	0.71	0.97	0.82	1445
1	0.11	0.01	0.02	589
accuracy			0.69	2034
macro avg	0.41	0.49	0.42	2034
weighted avg	0.53	0.69	0.59	2034

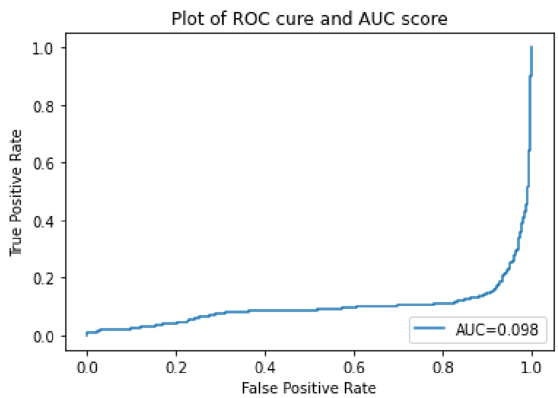




3.1.4 Model summary for b_conditional as training set and a_conditional as testing set

	precision	recall	f1-score	support
0	0.71	1.00	0.83	1359
1	0.00	0.00	0.00	548
accuracy			0.71	1907
macro avg	0.36	0.50	0.42	1907
weighted avg	0.51	0.71	0.59	1907





The figures in section 3.1.3 and 3.1.4 above provides the svc model summary for a_condiitional and b_conditional datasets. The confusion matrix shows that when b_conditional was used as the training set, the model predicted only one class which is not the case when a_condiitional is used. The model summary from both models also shows that when a_conditional was used as the training dataset, the model achieved a higher model performance in auc score and f1-score but not accuracy score which is the opposite when b_conditional was used as the training dataset. The auc score and roc curve for the model with b_conditional as training set is very poor. Although several techniques were applied by changing model parameters to improve performance, the result didn't get any better.

3.2 SVM classifier with GridSearchCV and hyperparameter tuning

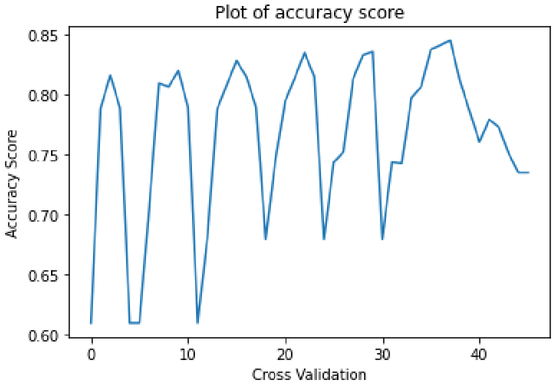
GridSearchCV is a technique in machine learning used to search through a grid of model parameters in order to obtain the best parameter for the model which will be used to make predictions. For this coursework, the GridSearchCV method in sklearn library was used with five-fold cross validation. Cross-validation is a resampling technique used to evaluate machine learning models on a limited dataset. The python code below implements the SVM classifier with a GridSearchCV method.

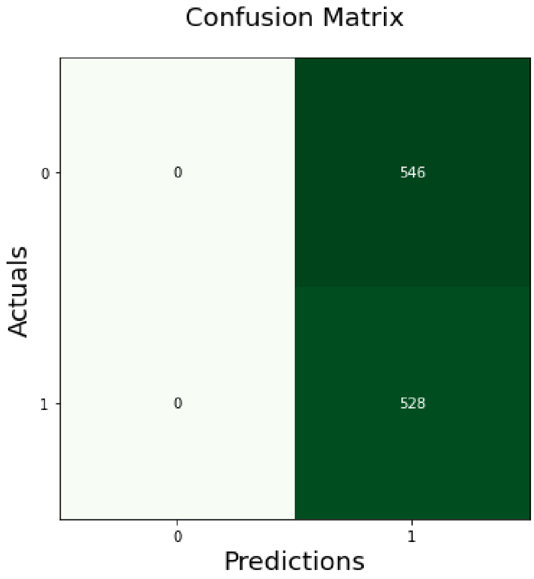
```
# scale the various dataframe using minmax scaler
sc = MinMaxScaler()
# defining turning parameter
tuned_parameters_1 = [{"kernel": ["rbf"], "gamma": [10, 1, 0.1, 0.01, 0.001, 0.0001],
                       "C": [0.1, 1, 10, 100, 1000,10000]},
    {"kernel": ["linear"], "C": [1, 10, 100, 1000,10000]},
    {"kernel": ["poly"], "C": [1, 10, 100, 1000,10000]},]

# Gridsearch cv and model fit for the a_affirmative_sc dataset use as trianing data set
clf = GridSearchCV(SVC(probability=True), tuned_parameters_1, refit = True, scoring="accuracy", cv = 5)
```

```
# function for model building
def model_fun(train_x,train_y,test_x,test_y):

  # standardised data
  test_x_sc = sc.fit_transform(test_x)
  train_x_sc = sc.fit_transform(train_x)

  # fit the model
  model = clf.fit(train_x_sc, train_y)

  #print best model parameter
  print("Best model parameters:",model.best_params_)
  #print best model score
  print("Best model score:",model.best_score_)

  #plot accuracy score
  cv_scores = model.cv_results_["mean_test_score"]
  plt.plot(cv_scores)
  plt.ylabel('Accuracy Score')
  plt.xlabel('Cross Validation')
  plt.title("Plot of accuracy score")
  plt.show()

  # model prediction and confusion matrix
  y_pred = model.predict(test_x)
  # plot confusion matrix for the model prediction
  fig, ax = plot_confusion_matrix(conf_mat=confusion_matrix(test_y,y_pred), figsize=(6, 6), cmap=plt.cm.Greens)
  plt.xlabel('Predictions', fontsize=18)
  plt.ylabel('Actuals', fontsize=18)
  plt.title('Confusion Matrix', fontsize=18)
  plt.show()
  print(classification_report(test_y, y_pred))

  # creating ROC and AUC for the model
  #define metrics
  y_pred_proba = model.predict_proba(test_x_sc)[::,1]
  fpr, tpr, _ = metrics.roc_curve(test_y,  y_pred_proba)
  auc = metrics.roc_auc_score(test_y, y_pred_proba)
```

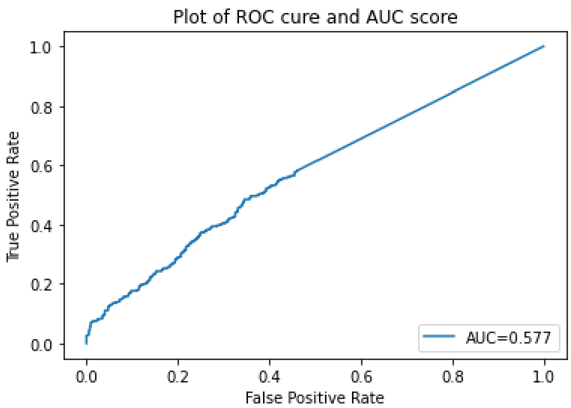
3.2.1 Model summary for a_affirmative as training set and b_affirmative as testing set

```
Best model parameters: {'C': 10, 'kernel': 'linear'}
Best model score: 0.8455000442909025
```



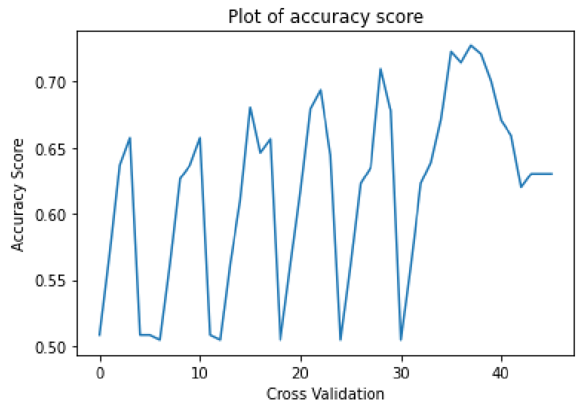


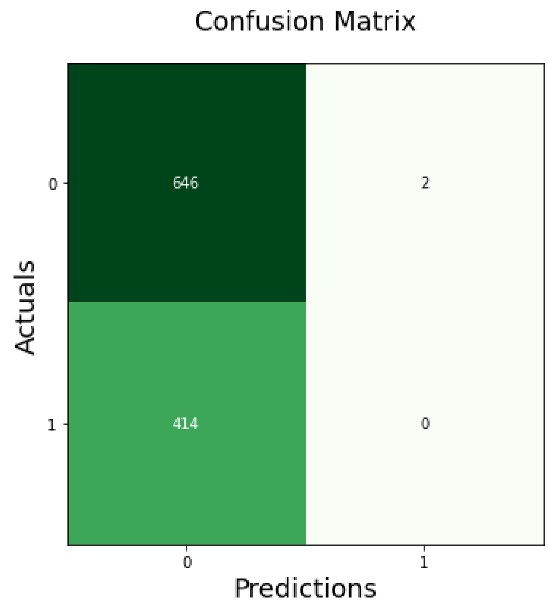

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       546
           1       0.49      1.00      0.66       528

    accuracy                           0.49      1074
   macro avg       0.25      0.50      0.33      1074
weighted avg       0.24      0.49      0.32      1074
```



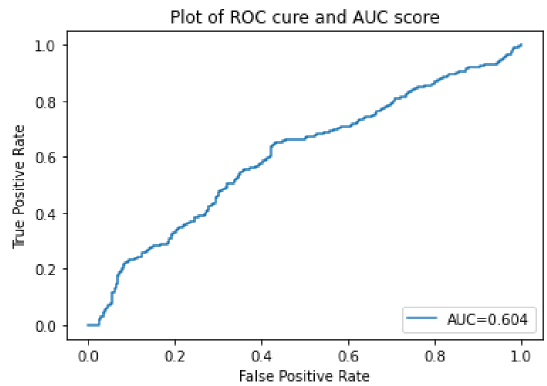
3.2.2 Model summary for b_affirmative as training set and a_affirmative as testing set

```
Best model parameters: {'C': 10, 'kernel': 'linear'}
Best model score: 0.7270941099760921
```





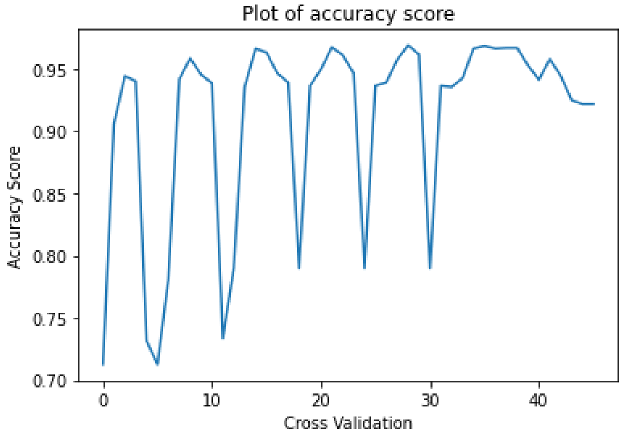

```
              precision    recall  f1-score   support

           0       0.61      1.00      0.76       648
           1       0.00      0.00      0.00       414

    accuracy                           0.61      1062
   macro avg       0.30      0.50      0.38      1062
weighted avg       0.37      0.61      0.46      1062
```



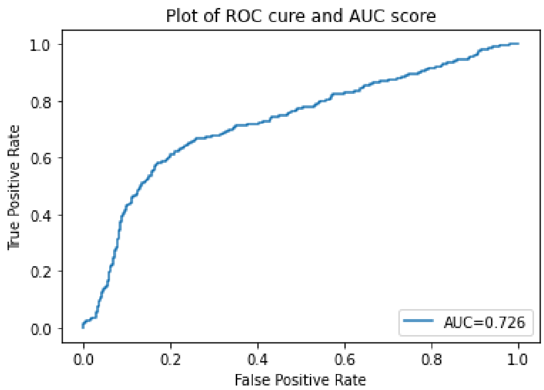
Section 3.2.1 and 3.2.2 above provide the model summary for the svc model with gridsearchcv for a_affirmative and b_affirmative dataset. The model with a_affirmative as training data achieved a better performance on the training set than The model with b_affirmative as training data. Both models suggested the same model parameters but the model with b_affirmative as training data showed a better model performance on the test set (a_affirmative) in all three performance metrics considered for this study. This better performance can be attributed to the fact that the b_affirmative dataset suffers less from class imbalance than the a_affirmative dataset.

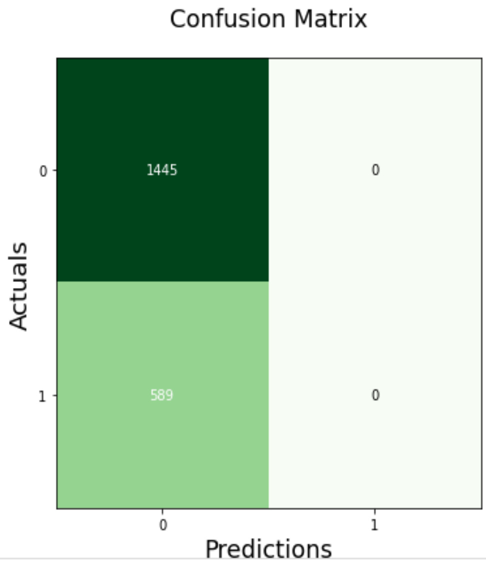
3.2.3 Model summary for a_conditional as training set and b_conditional as testing set

```
Best model parameters: {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
Best model score: 0.9690618515617485
```



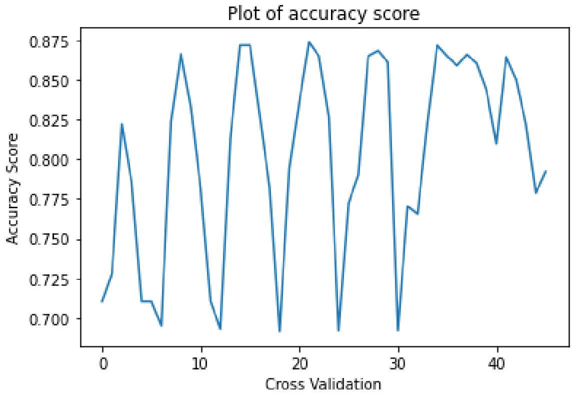
	precision	recall	f1-score	support
0	0.71	1.00	0.83	1445
1	0.00	0.00	0.00	589
accuracy			0.71	2034
macro avg	0.36	0.50	0.42	2034
weighted avg	0.50	0.71	0.59	2034



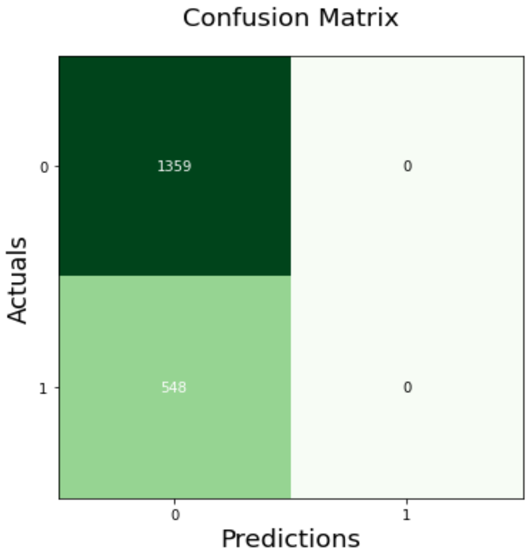


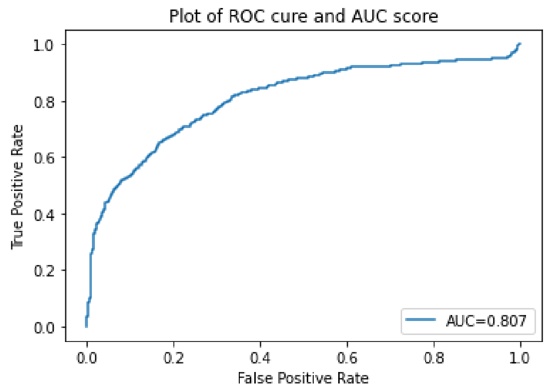
3.2.4 Model summary for b_conditional as train set and a_conditional as test set

```
Best model parameters: {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
Best model score: 0.8736011425666597
```



	precision	recall	f1-score	support
0	0.71	1.00	0.83	1359
1	0.00	0.00	0.00	548
accuracy			0.71	1907
macro avg	0.36	0.50	0.42	1907
weighted avg	0.51	0.71	0.59	1907



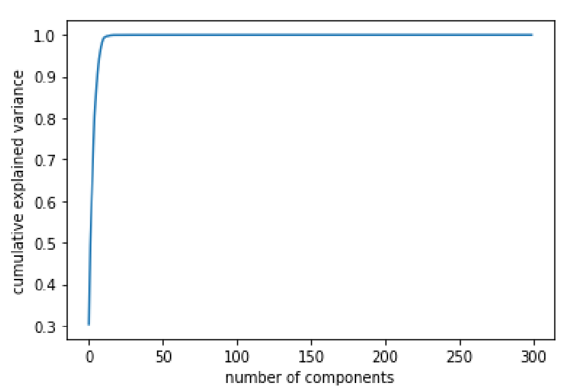


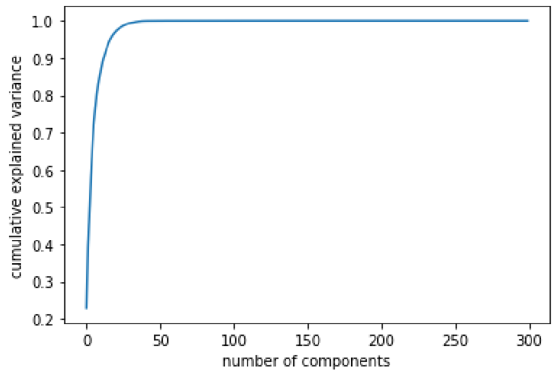
Section 3.2.3 and 3.2.4 above provide the model summary for the svc model with gridsearchcv for a_conditional and b_conditional dataset. The model with a_conditional as training data achieved a better model performance on the training set than The model with b_conditional as training data. Both models predicted only one class and achieved the same accuracy and f-1 score on their test dataset. But the model with b_conditional as training set achieved a better auc score.

3.3 Principal Component Analysis and SVM

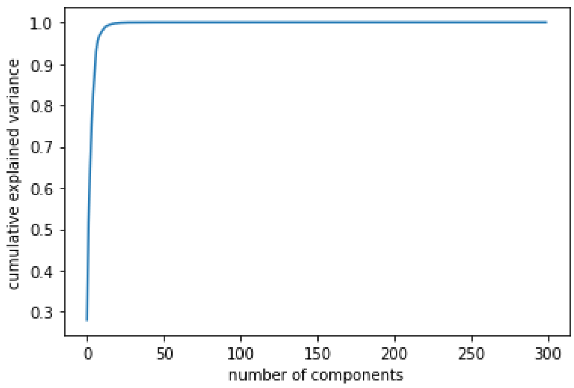
In this section, PCA with twenty components was used to extract a different feature representation other than the original landmark coordinate vector in the dataset and a linear SVM classifier was built to model the data.

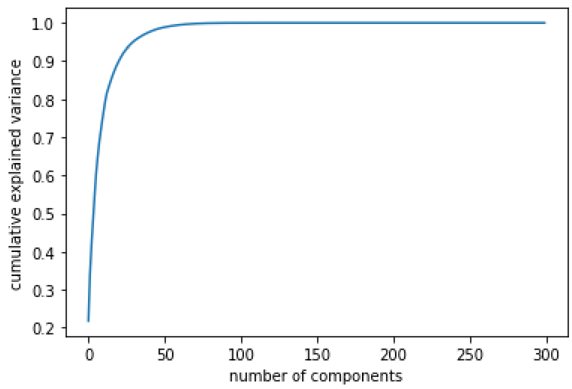
3.3.1 Cumulative proportion of variance explained for a_affirmative and b_affirmative



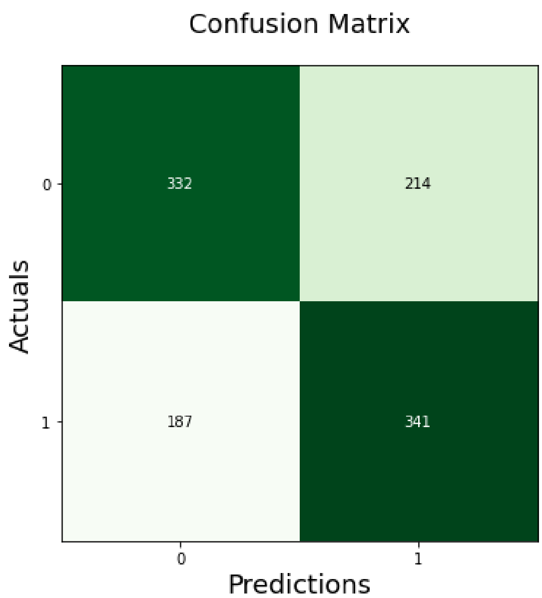


3.3.2 Cumulative proportion of variance explained for a_conditional and b_conditional

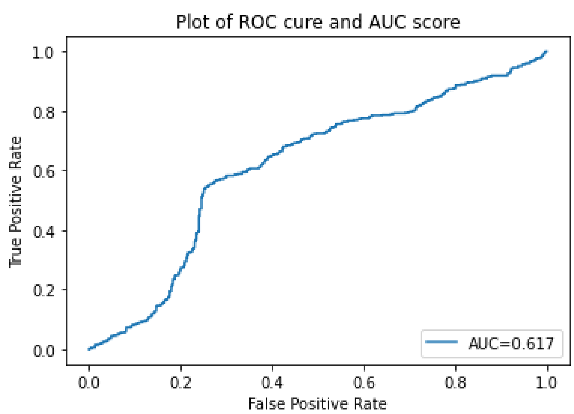




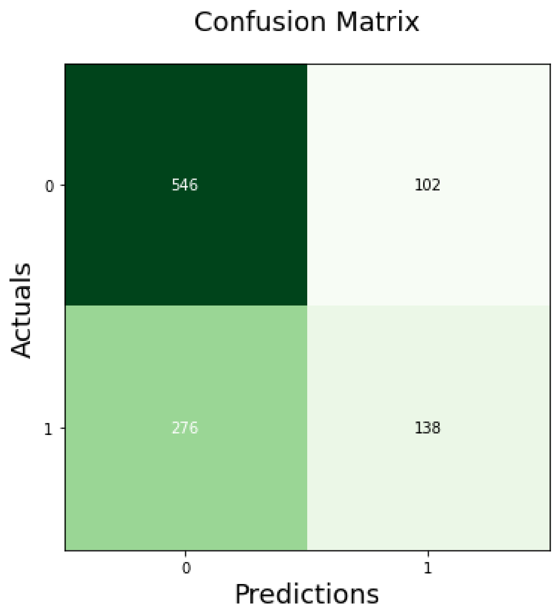
3.3.3 Model summary for a_affirmative as training set and b_affirmative as testing set



	precision	recall	f1-score	support
0	0.64	0.61	0.62	546
1	0.61	0.65	0.63	528
accuracy			0.63	1074
macro avg	0.63	0.63	0.63	1074
weighted avg	0.63	0.63	0.63	1074

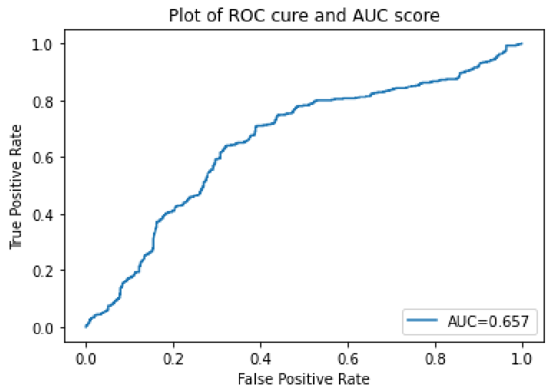


3.3.4 Model summary for b_affirmative as training set and a_affirmative as testing set

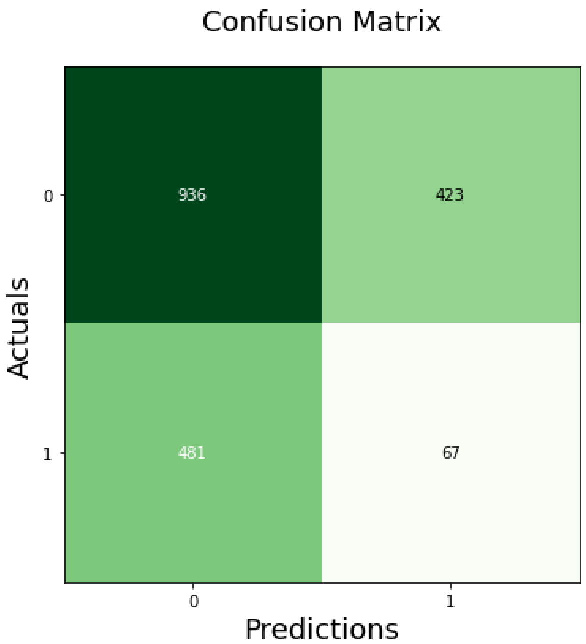


Predictions

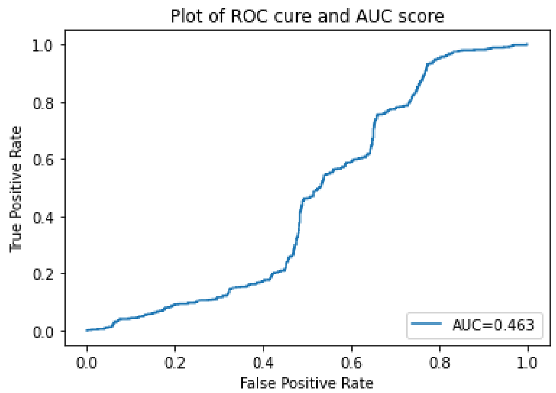
	precision	recall	f1-score	support
0	0.66	0.84	0.74	648
1	0.57	0.33	0.42	414
accuracy			0.64	1062
macro avg	0.62	0.59	0.58	1062
weighted avg	0.63	0.64	0.62	1062



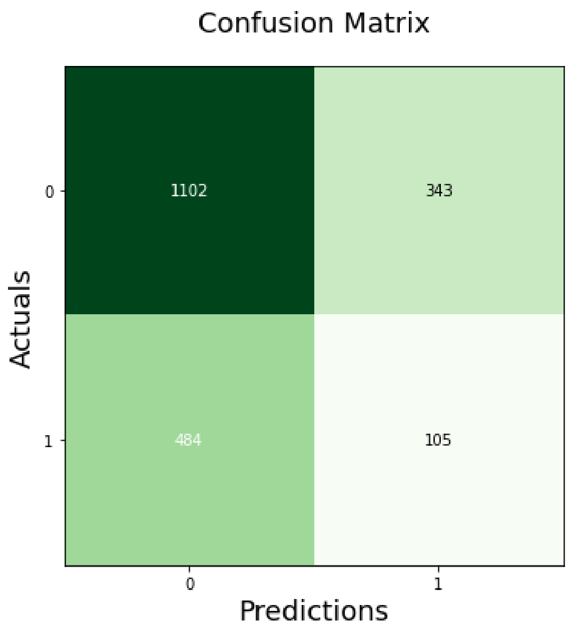
3.3.5 Model summary for a_conditional as training set and b_conditional as testing set



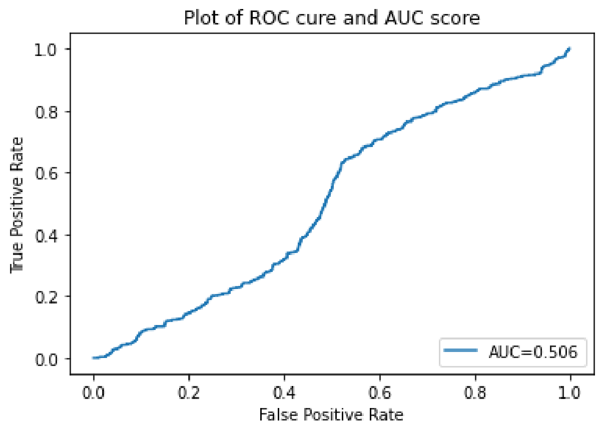
	precision	recall	f1-score	support
0	0.66	0.69	0.67	1359
1	0.14	0.12	0.13	548
accuracy			0.53	1907
macro avg	0.40	0.41	0.40	1907
weighted avg	0.51	0.53	0.52	1907



3.3.6 Model summary for b_conditional as training set and a_conditional as testing set



	precision	recall	f1-score	support
0	0.69	0.76	0.73	1445
1	0.23	0.18	0.20	589
accuracy			0.59	2034
macro avg	0.46	0.47	0.46	2034
weighted avg	0.56	0.59	0.58	2034



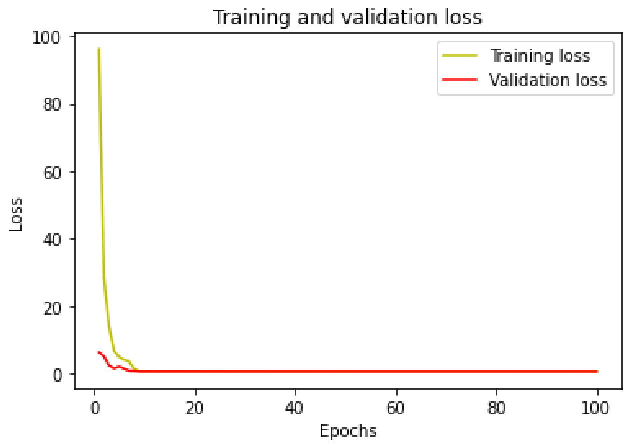
Section 3.3.3, 3.3.4, 3.3.5 and 3.36 above provide the model summary for the SVM algorithm with PCA model for a_ffirmative, b_affirmative, a_conditional and b_conditional dataset. All models achieved an accuracy over 0.5 and do not predict only one particular class as seen in section 3.1 and 3.2. The model with b_affirmative as training set achieved the best model performance among other models using SVC with PCA.

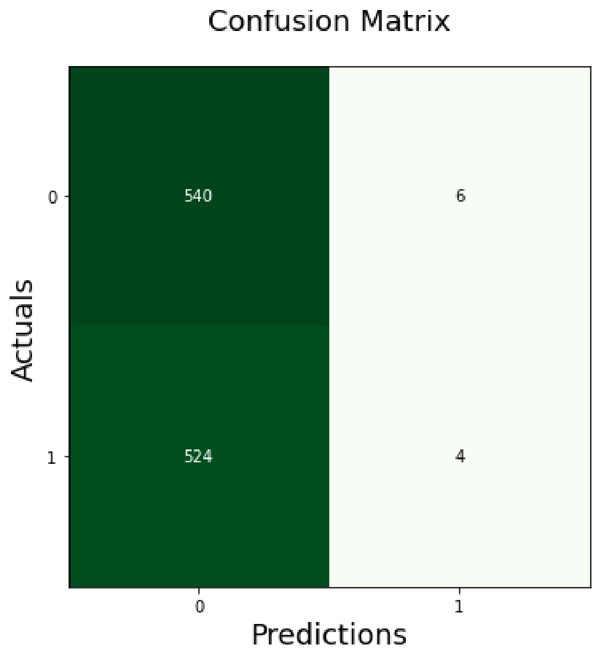
4.0 Artificial Neural Network Model

Artificial neural network (ANN) in recent times has become a very popular algorithm for model classification, regression, pattern recognition and prediction within the machine learning community [5]. This is because of its capability in achieving a high performance rating over traditional regression and statistical models. The ANN algorithm which works similarly to the biological nervous system of the human brain has a wide range of applications which includes image recognition, natural language processing and so on [6]. ANN algorithm is very effective and efficient in providing a high level capability in handling complex and non-complex tasks in several domains ranging from medical sciences, education, finance, engineering, security and manufacturing. In this coursework, a fully connected ANN algorithm with four dense layers and three dropout layers was implemented. The python code below implements the ANN algorithms.

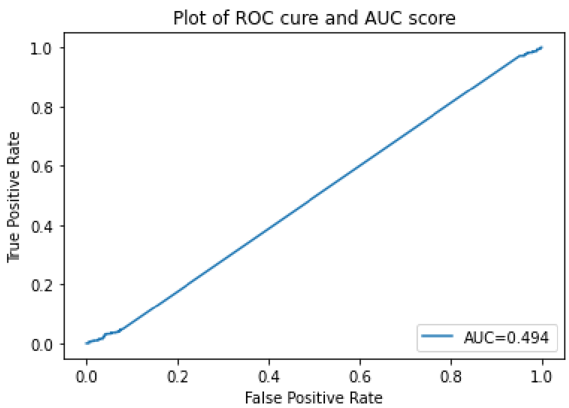
```
def ANN(train_x,train_y,test_x,test_y):
    # code to implement ANN algorithm
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=5)
    model = keras.Sequential([
    keras.layers.Dense(300, input_shape=(len(train_x.columns),), activation = 'relu'),
    keras.layers.Dense(200, activation='relu'),
    keras.layers.Dropout(0.25),
    keras.layers.Dense(100, activation ='relu'),
    keras.layers.Dropout(0.30),
    keras.layers.Dense(1, activation='sigmoid'),])
    # code to compile model
    model.compile(optimizer='adam',loss='binary_crossentropy',
                  metrics =['accuracy'])
    # cdoe to print model summary
    print(model.summary())
    # code to fit the model
    history = model.fit(train_x,train_y,batch_size=50,epochs=100, validation_split = 0.2,
                        verbose=2)
    #plot training and validation loss at each epoch
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(loss)+1)
    plt.plot(epochs, loss,'y',label ='Training loss')
    plt.plot(epochs,val_loss,'r', label = 'Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    # code to predict with test data
    pred = model.predict(test_x)
    # plot confusion matrix for the model prediction
    fig, ax = plot_confusion_matrix(conf_mat=confusion_matrix(test_y,np.round(pred)), figsize=(6, 6), cmap=plt.cm.Greens)
    plt.xlabel('Predictions', fontsize=18)
    plt.ylabel('Actuals', fontsize=18)
    plt.title('Confusion Matrix', fontsize=18)
    plt.show()
    print(classification_report(test_y, np.round(pred)))
    fpr, tpr, _ = metrics.roc_curve(test_y, pred)
    auc = metrics.roc_auc_score(test_y, pred)
```

4.1 Model summary for a_affirmative as training set and b_affirmative as testing set

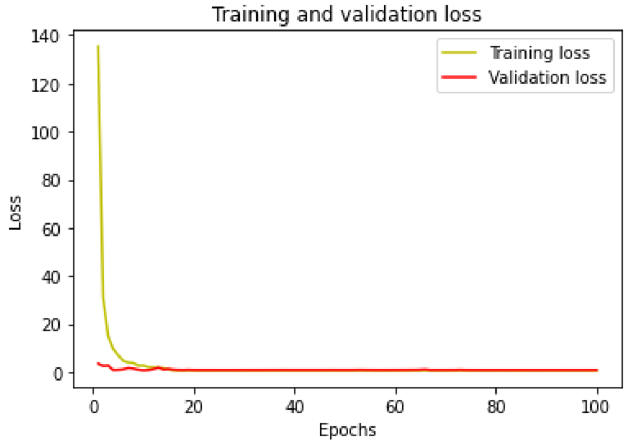


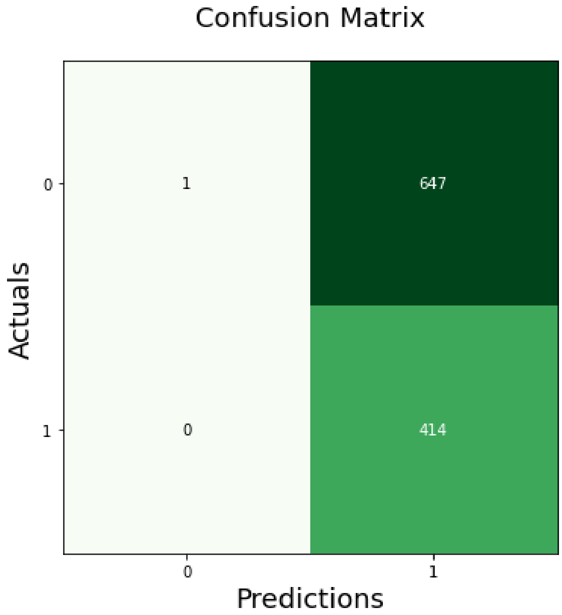


	precision	recall	f1-score	support
0	0.51	0.99	0.67	546
1	0.40	0.01	0.01	528
accuracy			0.51	1074
macro avg	0.45	0.50	0.34	1074
weighted avg	0.45	0.51	0.35	1074

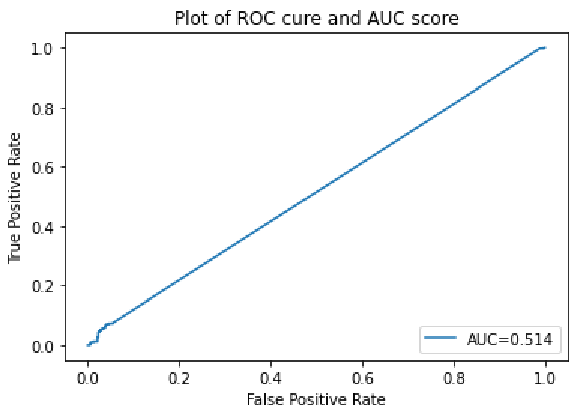


4.2 Model summary for b_affirmative as training set and a_affirmative as testing set



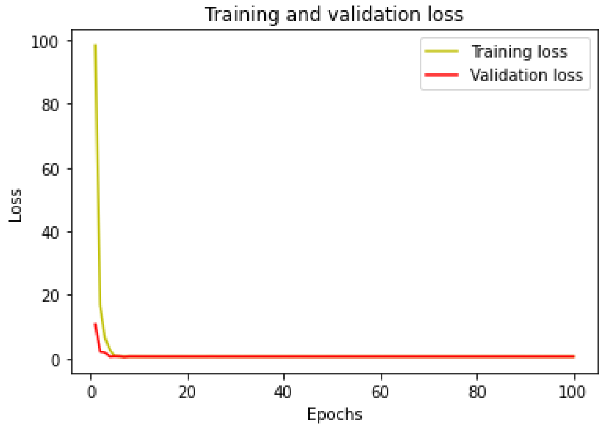


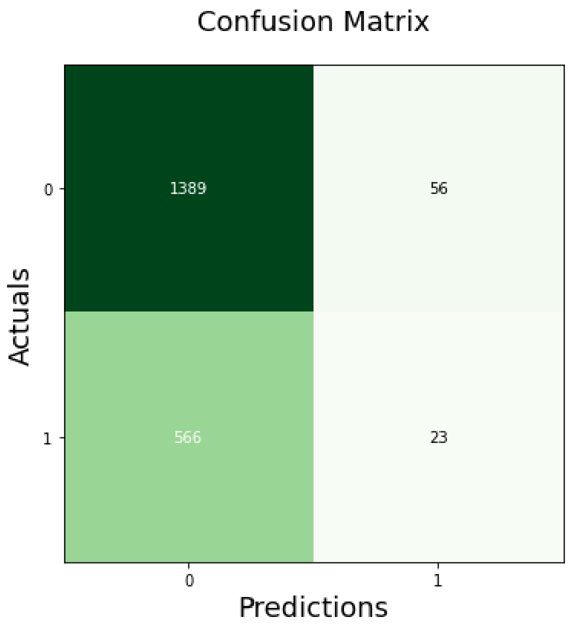
	precision	recall	f1-score	support
0	1.00	0.00	0.00	648
1	0.39	1.00	0.56	414
accuracy			0.39	1062
macro avg	0.70	0.50	0.28	1062
weighted avg	0.76	0.39	0.22	1062



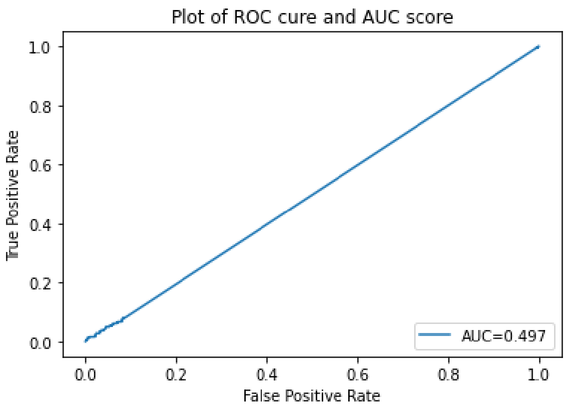
Section 4.1 and 4.2 above provide the model summary for the ANN model for a_affirmative and b_affirmative dataset. The ANN model with a_affirmative as training dataset achieved a better model performance in terms of accuracy score and f1-score while the ANN model with b_affirmative as training dataset achieved auc score but poor accuracy score and f1-score.

4.3 Model summary for a_conditional as training set and b_conditional as testing set

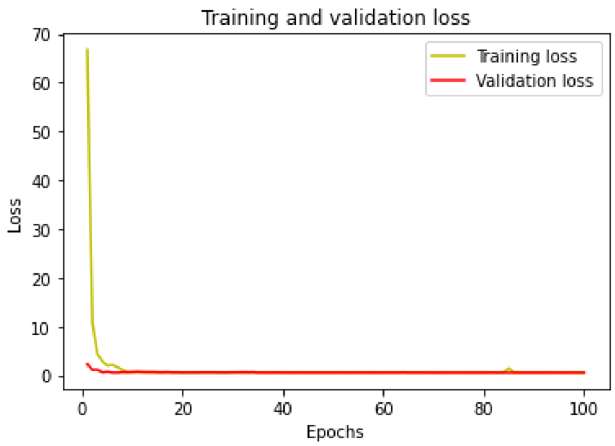


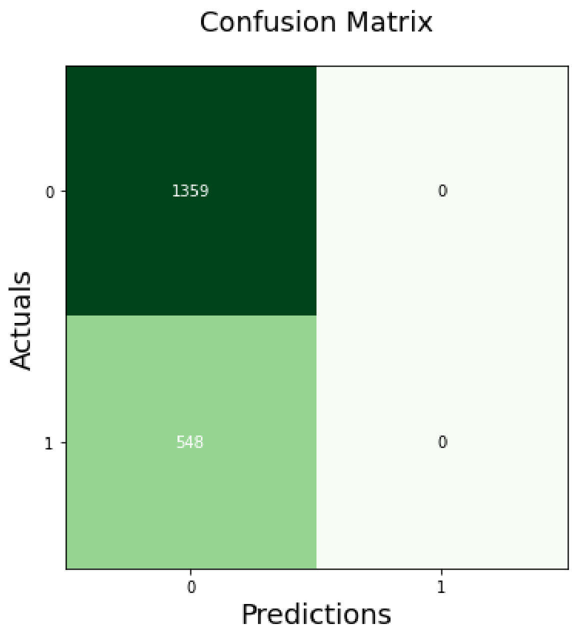


	precision	recall	f1-score	support
0	0.71	0.96	0.82	1445
1	0.29	0.04	0.07	589
accuracy			0.69	2034
macro avg	0.50	0.50	0.44	2034
weighted avg	0.59	0.69	0.60	2034

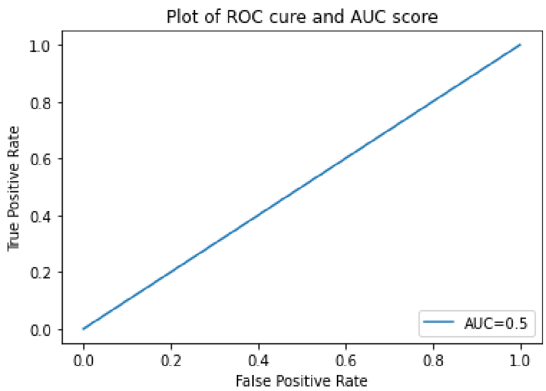


4.4 Model summary for b_conditional as training set and a_conditional as testing set






```
              precision    recall  f1-score   support

           0       0.71      1.00      0.83      1359
           1       0.00      0.00      0.00       548

    accuracy                           0.71      1907
   macro avg       0.36      0.50      0.42      1907
weighted avg       0.51      0.71      0.59      1907
```



Section 4.3 and 4.4 above provides the model summary of the ANN model for a_conditional and b_conditional dataset. The ANN model with b_conditional as training dataset achieved a slightly better model performance in all three performance metrics than the ANN model with b_conditional as training dataset. A major draw to the ANN model with b_conditional as training dataset is that the model predicts only one class which does not make the model adequate for generalisation.

5.0 Conclusion

This coursework provides an exhaustive experimentation with two machine learning classifiers (SVM and ANN) on two Grammatical facial expressions dataset (affirmative and conditional). Accuracy score was considered as primary performance metric for selecting the optimal model while AUC/ROC and F1-scores are considered as secondary performance metrics. The optimal algorithms were used to get predicted probabilities on the test set which are used to calculate the evaluation metrics summarised in the Table 5.1 below.

5.1 Model Summary

Model	Performance metrics			Dataset	
	Accuracy score	F-1 score	AUC	Training set	Test set
ANN	0.51	Class 0: 0.67 Class 1: 0.01	0.494	a_affirmative	b_affirmative
	0.39	Class 0: 0.00 Class 1: 0.56	0.514	b_affirmative	a_affirmative
	0.69	Class 0: 0.82 Class 1: 0.07	0.497	a_conditional	b_conditional
	0.71	Class 0: 0.83 Class 1: 0.00	0.5	b_conditional	a_conditional
Soft SVM without GridSearchCV	0.51	Class 0: 0.67 Class 1: 0.00	0.721	a_affirmative	b_affirmative
	0.39	Class 0: 0.00 Class 1: 0.56	0.465	b_affirmative	a_affirmative
	0.69	Class 0: 0.82 Class 1: 0.02	0.62	a_conditional	b_conditional
	0.71	Class 0: 0.83 Class 1: 0.00	0.098	b_conditional	a_conditional
SVM with GridSearchCV	0.49	Class 0: 0.00 Class 1: 0.66	0.577	a_affirmative	b_affirmative
	0.61	Class 0: 0.76 Class 1: 0.00	0.604	b_affirmative	a_affirmative
	0.71	Class 0: 0.83 Class 1: 0.00	0.728	a_conditional	b_conditional
	0.71	Class 0: 0.83 Class 1: 0.00	0.807	b_conditional	a_conditional
SVM with PCA	0.63	Class 0: 0.62 Class 1: 0.63	0.617	a_affirmative	b_affirmative
	0.64	Class 0: 0.74 Class 1: 0.42	0.657	b_affirmative	a_affirmative
	0.53	Class 0: 0.67 Class 1: 0.13	0.463	a_conditional	b_conditional
	0.59	Class 0: 0.73 Class 1: 0.20	0.506	b_conditional	a_conditional

From table 5.1 above, ANN algorithm and Soft SVM algorithm achieved the same accuracy for all the case studies considered in this coursework while the SVM with GridSearchCV algorithm achieved a higher accuracy than ANN and soft SVM in ¾ cases considered in this coursework. The SVM with PCA model achieved an accuracy score of over 50% in all cases. From the results above, the SVM with

GridSearchCV and the SVM models with PCA is perceived to be a better model for generalisation as they do not predict only a particular class in all the cases in studies as seen in order model predictions.

A major limitation encountered in this coursework are the class imbalance inherent in the dataset and limitation in the size of dataset in each case study. Further strategy to improve performance is to collect a more balanced dataset in order to build better and efficient classification models. From our study, all our model performances when compared to the reference model published in [7][8] achieved a lower performance. This is because computing several SVM and ANN models together with hyperparameter tuning is computationally expensive and time expensive. Which is a major limitation encountered in this coursework.

Appendix

Link to Python Code in Google Colab:
https://colab.research.google.com/drive/1yBswrssqbyJFdabMBlgdu2cucEsaySv8?usp=sharing

Reference

[1] Gharib, M., Lollini, P., Botta, M., Amparore, E., Donatelli, S. and Bondavalli, A., 2018, June. On the safety of automotive systems incorporating machine learning based components: a position paper. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)* (pp. 271-274). IEEE.

[2] Bishop, C.M., 2006. Pattern recognition. *Machine learning*, *128*(9).

[3] Seliya, N., Khoshgoftaar, T. M. and Hulse, J. V. 'A Study on the Relationships of Classifier Performance Metrics'. 2009 21st IEEE International Conference on Tools with Artificial Intelligence, 2–4 Nov. 2009, 59–66.

[4] Vapnik Vladimir, N., 1995. The nature of statistical learning theory.

[5] Dave, V.S. and Dutta, K., 2014. Neural network based models for software effort estimation: a review. *Artificial Intelligence Review*, *42*(2), pp.295-307.

[6] Wang, D., He, H. and Liu, D., 2017. Adaptive critic nonlinear robust control: A survey. *IEEE transactions on cybernetics*, *47*(10), pp.3429-3451.

[7] Bhuvan, M.S., Rao, D.V., Jain, S., Ashwin, T.S., Guddetti, R.M.R. and Kulgod, S.P., 2016, May. Detection and analysis model for grammatical facial expressions in sign language. In *2016 IEEE Region 10 Symposium (TENSYMP)* (pp. 155-160). IEEE.

[8] de Almeida Freitas, F., Peres, S.M., de Moraes Lima, C.A. and Barbosa, F.V., 2014, May. Grammatical facial expressions recognition with machine learning. In *The Twenty-seventh international flairs conference*.