Fin: Prediction of corporate finance incidents

Piyush Rane, Raj Patil, Swanand Barsawade
RED ID: 826159835, 825897040, 825876097
Computer Science Department
San Diego State University
San Diego, USA
prane2174@sdsu.edu, rpatil0722@sdsu.edu, sbarsawade0801@sdsu.edu

***Abstract*—Machine learning algorithms can be used in building predictive models which is helpful in taking business decisions. In this project we have developed machine learning models for predicting whether a company will be acquired based on its historical financial data. We have developed sequential as well non sequential models for predicting the output values. In sequential model we have used Logistic Regression, Support Vector Machine and Feed Forward Neural Network. Among these models support vector machine had low accuracy of 96 percent whereas other models had accuracy of 99 percent. But, the precision, recall and the f1 score of SVM is good as compared to other non-sequential models. In sequential model we have used LSTM (Long short-term memory) and GRU (Gated Recurrent Unit) and where we predicted whether the company will get acquired based on it's previous year's data. We found out that accuracy of about 90 percent for LSTM and 75 percentage for GRU models. LSTM is found out to be a better model as compared to GRU model in terms of accuracy and AUC score.**

I. Introduction

Leveraging the power of machine learning organization can now take efficient business decisions for their growth of organization. Machine learning models can also be used in predicting whether the company will get acquired based on it's previous data or not. With this, new startups and small size companies can use such models to predict whether they can get acquired by other big companies based on it's past performance data. In this project different features like total assets of firm, net profit of firm, total revenue of the firm etc are the factors that has impact whether the company will get acquired or not. In this data set the target variable is named as target having value as 0 indicating that company is not acquired and 1 indicating the company is acquired. Overall it has total 16 features and 1 target feature. In total there are around 225010 records. For this task we have built sequential and non sequential models. Non sequential models like logistic regression, Feedforward Neural Network and Support Vector Machine (SVM) are used in building predictive models. Metrics like F1-score, accuracy and ROC value were used to evaluate the performances of the model. In non sequential models we have used Long short-term memory and Gated Recurrent Unit(GRU) in which accuracy of LSTM turned out to be better than GRU model in terms of accuracy and AUC value. As the data set was highly imbalanced we have to clean the data set before working on it has it had many null values for different records. So, before building the models data cleaning operation was performed to get the accurate results.

II. Data Cleaning

A. Dropping blank columns

There were two blank columns ca and Unnamed:0 which were dropped as it did not contained any data.

B. Dropping records

There were records which had around more than half of the columns with values as NULL. So, we dropped such records which had more than half of the columns with values as NULL values. After dropping the columns we got resultant data set as 190345 records.

III. Non-sequential models

A. Logistic Regression

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X. Instead of fitting a straight line or hyperplane, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1.

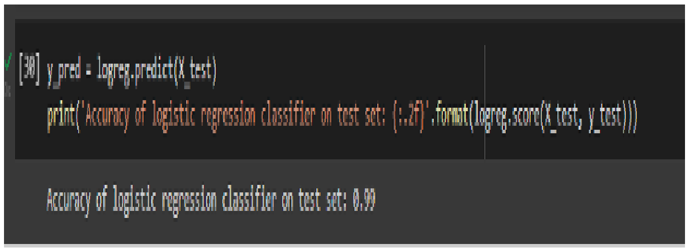
Logistic Regression Assumptions

- Binary logistic regression requires the dependent variable to be binary.
- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- The independent variables should be independent of each other. That is, the model should have little or no multicollinearity.
- The independent variables are linearly related to the log odds. Logistic regression requires quite large sample sizes.

Steps followed to build the model:

- We created a data set with 190345 rows and 13 columns. There is an unnamed column so we can not consider it for our study,thus we dropped it.
- We observed that there are multiple rows with NaN values. We dropped all the rows with more than 50 percent NaN values and imputed mean for the rest of the rows with less than 50 percent NaN values
- Analyzed the target variable and observed that the data set is inconsistent and biased as there are almost 99 percent of 0's and only 1 percent of 1's.
- Selected the dependent and independent variables from the data set implemented the model to check the p-values
- Dropped the variables with the p-value greater than 0.05.
- We split the data set into training and testing data frames. We assigned 30 percent of data set for testing and 70 percent for training and trained the model
- Then, we analyzed the model using different metrics.

Logistic Regression Accuracy:



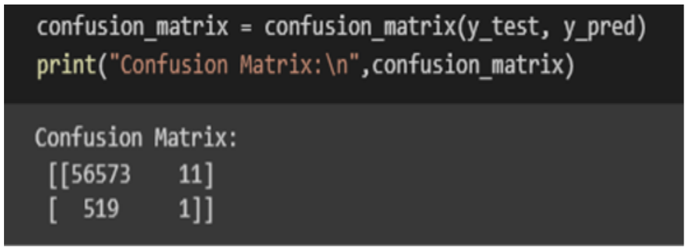
```
[30] y_pred = logreg.predict(X_test)
     print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))

Accuracy of logistic regression classifier on test set: 0.99
```

Fig. 1: Logistic Regression Accuracy.

Logistic Regression Accuracy comes out to be 99 percent.

Logistic Regression Confusion Matrix:



```
confusion_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n",confusion_matrix)

Confusion Matrix:
 [[56573    11]
 [  519     1]]
```

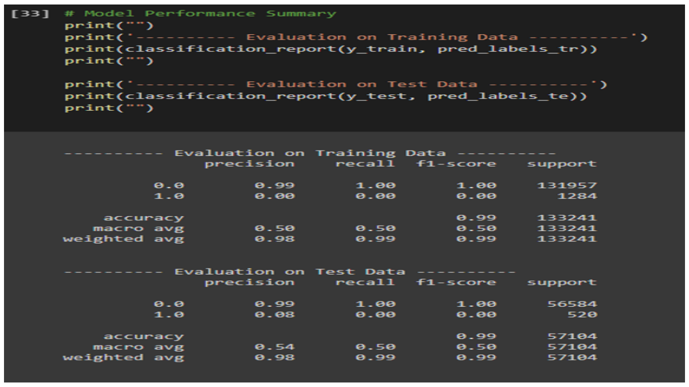
Fig. 2: Logistic Confusion Matrix.

The result is telling us that we have 56573+11 correct predictions and 519+1 incorrect predictions.

Summary and F-1 score:

We can see the F-1 score and the accuracy of our model on the training as well as the testing dataframes. The F-1 score is 1 which signifies that our model is able to perfectly classify each of the observations into a class.

Receiver operating characteristic curve (ROC)



```
[33] # Model Performance Summary
     print("")
     print('---------- Evaluation on Training Data ----------')
     print(classification_report(y_train, pred_labels_tr))
     print("")

     print('---------- Evaluation on Test Data ----------')
     print(classification_report(y_test, pred_labels_te))
     print("")

---------- Evaluation on Training Data ----------
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00    131957
         1.0       0.00      0.00      0.00      1284

    accuracy                           0.99    133241
   macro avg       0.50      0.50      0.50    133241
weighted avg       0.98      0.99      0.99    133241

---------- Evaluation on Test Data ----------
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00     56584
         1.0       0.08      0.00      0.00       520

    accuracy                           0.99     57104
   macro avg       0.54      0.50      0.50     57104
weighted avg       0.98      0.99      0.99     57104
```

Fig. 3: Logistic Regression summary and F-1 score.

ROC- We observe the ROC curve to be almost along the baseline. This might be because of the bias in the target variable.

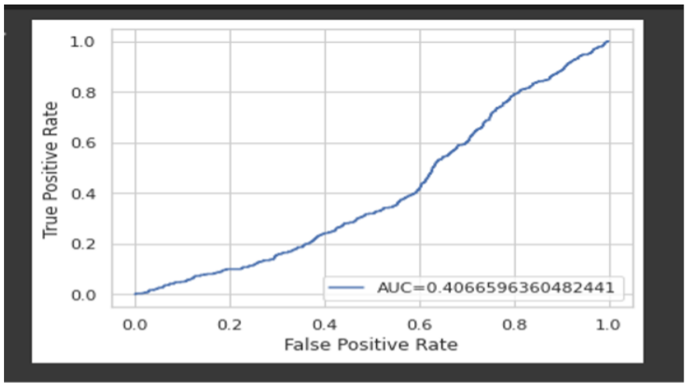


Fig. 4: Logistic Regression ROC

B. *Feedforward Neural Network*

Different layers of Feedforward Neural Network: //

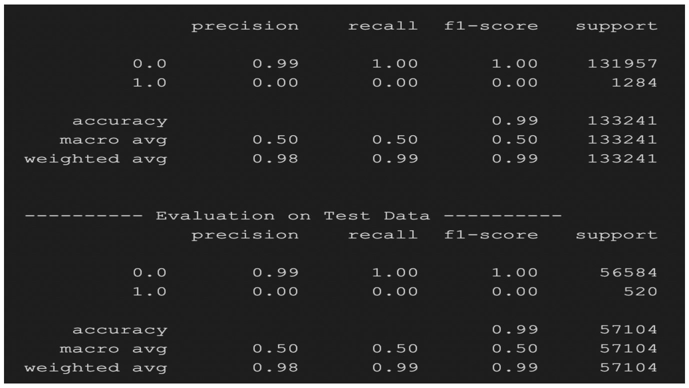
- Input Layer — contains one or more input nodes. For example, suppose you want to predict whether it will rain tomorrow and base your decision on two variables, humidity and wind speed. In that case, your first input would be the value for humidity, and the second input would be the value for wind speed.
- Hidden Layer — this layer houses hidden nodes, each containing an activation function (more on these later). Note that a Neural Network with multiple hidden layers is known as Deep Neural Network.
- Output Layer — contains one or more output nodes. Following the same weather prediction example above, you could choose to have only one output node generating a rain probability (where ¿0.5 means rain tomorrow, and 0.5 no rain tomorrow). Alternatively, you could have two output nodes, one for rain and another for no rain. Note, you can use a different activation function for output

nodes vs. hidden nodes. Connections — lines joining different nodes are known as connections. These contain kernels (weights) and biases, the parameters that get optimized during the training of a neural network

Steps followed to build the model:

- We created a dataset with 190345 rows and 13 columns. There is an unnamed column so we can not consider it for our study,thus we dropped it.
- We observed that there are multiple rows with NaN values. We dropped all the rows with more than 50 percent NaN values and imputed mean for the rest of the rows with less than 50 percent NaN values
- Analyzed the target variable and observed that the dataset is inconsistent and biased as there are almost 99 percent of 0's and only 1 percentof 1's.
- We split the dataset into training and testing dataframes. We assigned 30 percent of dataset for testing and 70 percent for training and trained the model
- We defined the structure of our neural network and compiled the model.
- We used 5 epochs to fit our model and analyzed the model using different metrics.

Feedforward Neural Network Summary:



```
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00    131957
         1.0       0.00      0.00      0.00      1284

    accuracy                           0.99    133241
   macro avg       0.50      0.50      0.50    133241
weighted avg       0.98      0.99      0.99    133241


---------- Evaluation on Test Data ----------
              precision    recall  f1-score   support

         0.0       0.99      1.00      1.00     56584
         1.0       0.00      0.00      0.00       520

    accuracy                           0.99     57104
   macro avg       0.50      0.50      0.50     57104
weighted avg       0.98      0.99      0.99     57104
```

Fig. 5: Feedforward Neural Network

We can see the F-1 score and the accuracy of our model on the training as well as the testing dataframes. The F-1 score is 1 which signifies that our model is able to perfectly classify each of the observations into a class and also accuracy is 99 percent.

Receiver operating characteristic curve (ROC): We can see that AUC score is approx 50 percent which is linear and not that great score for this model.

C. Support Vector Machine

In machine learning, support-vector machines (SVMs, also support-vector networks[1]) are supervised learning models

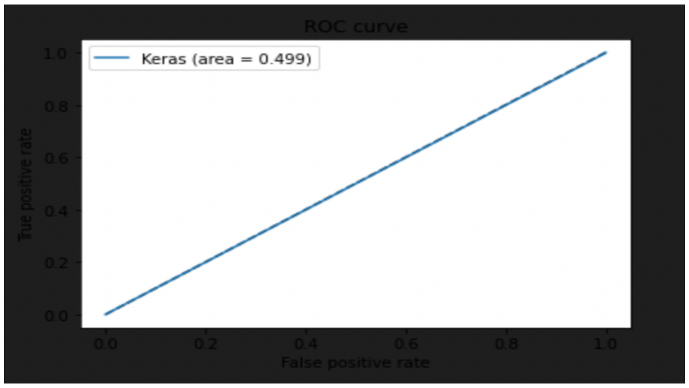
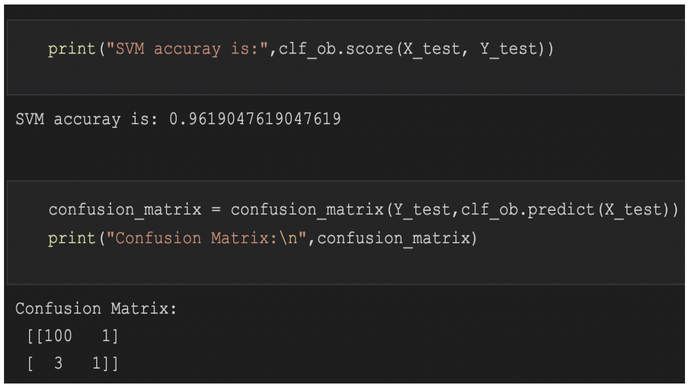


Fig. 6: Feedforward Neural Network - ROC

with associated learning algorithms that analyze data for classification and regression analysis. SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

SVM confusion matrix and accuracy:



```
print("SVM accuray is:",clf_ob.score(X_test, Y_test))
```

```
SVM accuray is: 0.9619047619047619
```

```
confusion_matrix = confusion_matrix(Y_test,clf_ob.predict(X_test))
print("Confusion Matrix:\n",confusion_matrix)
```

```
Confusion Matrix:
 [[100   1]
 [  3   1]]
```

Fig. 7: SVM - Confusion Matrix and Accuracy

From above snapshot we can say that it has less accuracy which is of 96 percent as compared to other non-sequential models.

SVM Receiver operating characteristic curve (ROC): The AUC score of SVM is 48 percent which is quite good as compared to Feedforward neural network.

SVM classification report : It can be seen that SVM has overall good f1, precision and recall score but has less accuracy as compared to other models.

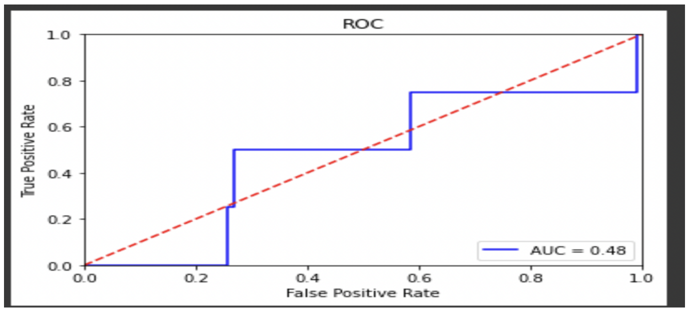
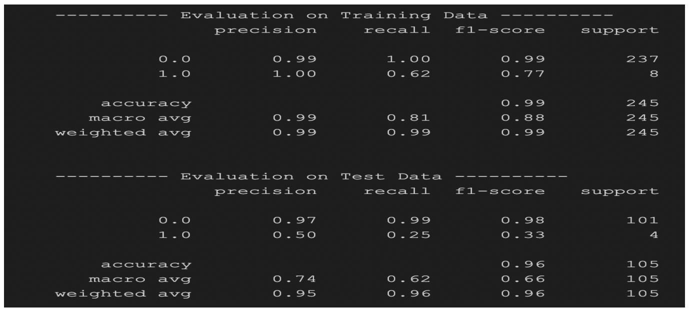


Fig. 8: SVM - ROC



```
---------- Evaluation on Training Data ----------
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99       237
         1.0       1.00      0.62      0.77         8

    accuracy                           0.99       245
   macro avg       0.99      0.81      0.88       245
weighted avg       0.99      0.99      0.99       245

---------- Evaluation on Test Data ----------
              precision    recall  f1-score   support

         0.0       0.97      0.99      0.98       101
         1.0       0.50      0.25      0.33         4

    accuracy                           0.96       105
   macro avg       0.74      0.62      0.66       105
weighted avg       0.95      0.96      0.96       105
```

Fig. 9: SVM - Classification Report

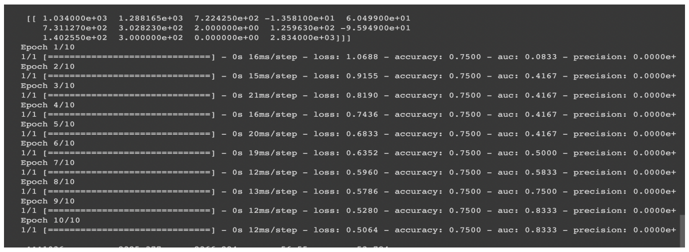
IV. Sequential models

A. *GRU (Gated Recurrent Unit)*

In a nutshell, GRUs are an improved version of standard recurrent neural networks. They can also be considered as a variation of LSTM as the design for both these models is very similar. They work on a simple concept of update gate and reset gate, which are nothing but two vectors that take the call whether to pass certain information to the output or not. Their peculiarity is that they can be trained to keep information from long ago, without washing it through time or removing information which is irrelevant to the prediction. This is the very reason they offer improvements over LSTM and have simpler architecture as well.

Above figure is the result for one of the LSTM batches. LSTM ran for 10 epochs and accuracy remains the same for every epoch which is 90 percent but the AUC value increases from 70 percent to 95 percent.

GRU Result:



```
 [[ 1.034000e+03  1.288165e+03  7.224250e+02 -1.358100e+01  6.049900e+01
    7.311270e+02  3.028230e+02  2.000000e+00  1.259630e+02 -9.594900e+01
    1.402550e+02  3.000000e+02  0.000000e+00  2.834000e+03]]]
Epoch 1/10
1/1 [==============================] - 0s 16ms/step - loss: 1.0688 - accuracy: 0.7500 - auc: 0.0833 - precision: 0.0000e+
Epoch 2/10
1/1 [==============================] - 0s 15ms/step - loss: 0.9155 - accuracy: 0.7500 - auc: 0.4167 - precision: 0.0000e+
Epoch 3/10
1/1 [==============================] - 0s 21ms/step - loss: 0.8190 - accuracy: 0.7500 - auc: 0.4167 - precision: 0.0000e+
Epoch 4/10
1/1 [==============================] - 0s 16ms/step - loss: 0.7436 - accuracy: 0.7500 - auc: 0.4167 - precision: 0.0000e+
Epoch 5/10
1/1 [==============================] - 0s 20ms/step - loss: 0.6833 - accuracy: 0.7500 - auc: 0.4167 - precision: 0.0000e+
Epoch 6/10
1/1 [==============================] - 0s 19ms/step - loss: 0.6352 - accuracy: 0.7500 - auc: 0.5000 - precision: 0.0000e+
Epoch 7/10
1/1 [==============================] - 0s 12ms/step - loss: 0.5960 - accuracy: 0.7500 - auc: 0.5833 - precision: 0.0000e+
Epoch 8/10
1/1 [==============================] - 0s 13ms/step - loss: 0.5786 - accuracy: 0.7500 - auc: 0.7500 - precision: 0.0000e+
Epoch 9/10
1/1 [==============================] - 0s 12ms/step - loss: 0.5280 - accuracy: 0.7500 - auc: 0.8333 - precision: 0.0000e+
Epoch 10/10
1/1 [==============================] - 0s 12ms/step - loss: 0.5064 - accuracy: 0.7500 - auc: 0.8333 - precision: 0.0000e+
```

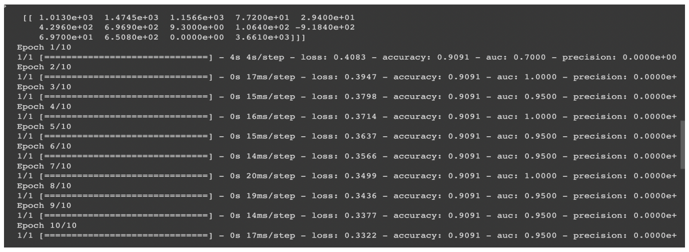
Fig. 10: Result for GRU.

Above figure is the result for one of the GRU batches. GRU ran for 10 epochs and accuracy remains the same for every epoch which is 75 percent but there is increase and decrease in AUC value which is same for first and tenth epoch 83 percent

B. *LSTM (Long -short-term memory)*

RNN consists of multiple layers similar to a Feed-Forward Neural Network: the input layer, hidden layer(s), and output layer. RNN contains recurrent units in its hidden layer, which allows the algorithm to process sequence data. It does it by recurrently passing a hidden state from a previous timestep and combining it with an input of the current one. The LSTM recurrent unit is much more complex than that of RNN, which improves learning but requires more computational resources. In a nutshell, LSTM is a special kind of recurrent neural network capable of handling long-term dependencies.

LSTM Result:



```
 [[ 1.0130e+03  1.4745e+03  1.1566e+03  7.7200e+01  2.9400e+01
    4.2960e+02  6.9690e+02  9.3000e+00  1.0640e+02 -9.1840e+02
    6.9700e+01  6.5080e+02  0.0000e+00  3.6610e+03]]]
Epoch 1/10
1/1 [==============================] - 4s 4s/step - loss: 0.4083 - accuracy: 0.9091 - auc: 0.7000 - precision: 0.0000e+00
Epoch 2/10
1/1 [==============================] - 0s 17ms/step - loss: 0.3947 - accuracy: 0.9091 - auc: 1.0000 - precision: 0.0000e+
Epoch 3/10
1/1 [==============================] - 0s 15ms/step - loss: 0.3798 - accuracy: 0.9091 - auc: 0.9500 - precision: 0.0000e+
Epoch 4/10
1/1 [==============================] - 0s 16ms/step - loss: 0.3714 - accuracy: 0.9091 - auc: 1.0000 - precision: 0.0000e+
Epoch 5/10
1/1 [==============================] - 0s 15ms/step - loss: 0.3637 - accuracy: 0.9091 - auc: 0.9500 - precision: 0.0000e+
Epoch 6/10
1/1 [==============================] - 0s 14ms/step - loss: 0.3566 - accuracy: 0.9091 - auc: 0.9500 - precision: 0.0000e+
Epoch 7/10
1/1 [==============================] - 0s 20ms/step - loss: 0.3499 - accuracy: 0.9091 - auc: 1.0000 - precision: 0.0000e+
Epoch 8/10
1/1 [==============================] - 0s 19ms/step - loss: 0.3436 - accuracy: 0.9091 - auc: 0.9500 - precision: 0.0000e+
Epoch 9/10
1/1 [==============================] - 0s 14ms/step - loss: 0.3377 - accuracy: 0.9091 - auc: 0.9500 - precision: 0.0000e+
Epoch 10/10
1/1 [==============================] - 0s 17ms/step - loss: 0.3322 - accuracy: 0.9091 - auc: 0.9500 - precision: 0.0000e+
```

Fig. 11: Result for LSTM.

Above figure is the result for one of the LSTM batches. LSTM ran for 10 epochs and accuracy remains the same for every epoch which is 90 percent but the AUC value increases from 70 percent to 95 percent. It can be seen that LSTM has better results than GRU.

C. *Common steps performed for GRU and LSTM*

The requirement wanted to use features from the previous years to predict whether the same company will be acquired in the next year.

So, to execute this task we performed following steps:

- Cleaned the data as described in the Data Cleaning Section.
- Created the batches of records based on companies. Each batch had company specific records.
- Each batch was then fed to LSTM and GRU models to get the results.

D. *Parameters used for LSTM and GRU*

After performing all the necessary operations for data cleaning, we chose the the correct layers and parameters for LSTM and GRU models like

- The input sequence has 1 values and running the example outputs a single value for the input sequence as a 2D array
- We set the activation function to use the default hyperbolic tangent (tanh)

- We set return_sequences to 'True' so as to to return the last output in the output sequence, or the full sequence
- An optimizer is one of the two arguments required for compiling a Keras model, so we instantiated an optimizer before passing it to model.compile() using the RMSprop algorithm. We used this for two reasons:
  - To maintain a moving (discounted) average of the square of gradients
  - To divide the gradient by the root of this average
- The purpose of loss functions is to compute the quantity that a model should seek to minimize during training, so we set it to the 'BinaryCrossentropy' class to compute the cross-entropy loss between true labels and predicted labels.
- Set the metrics parameter to accuracy so as to observe the accuracy of our models.

V. Conclusion and Future Work

It is very important to clean the data first before developing predictive models and uncleaned data can led to unreliable results. It can be observed that in sequential model SVM has accuracy of 96 percent which is less as compared to other models. But, we can see that precision, recall and F1 score are better than other models which makes it quite reliable model than other non-sequential model. Whereas, in sequential model we saw that LSTM;s accuracy and AUC score was higher as compared to GRU. So, in sequential model LSTM can be considered a relaible model when compared with GRU. As data set was highly unbalanced that result generated by the above mentioned model are not completely reliable. But, in future

VI. References

[1] https://scikit-learn.org/stable/ [2] https://pytorch.org/docs/stable/generated/torch.nn.RNN.html [3] https://en.wikipedia.org/wiki/Support-vector$_machine$