COMS 4995 - Applied Machine Learning

Project Deliverable #3 - Report

Objective

In this project, we intend to build an effective classification model to classify different types of 'attacks'. Since the NSL-KDD dataset has a huge imbalance in the target 'attack' variable, we used SMOTE on the pre-processed dataset to balance the dataset and stored training dataset in 'Oversampled_X_train.csv' and 'Oversampled_y_train.csv'.

ML models - Results & Implementation

1. Random Forest

```
Accuracy of training data: 1.0
Accuracy of testing data: 0.7657365922902897
```

	precision	recall	f1-score	support
Dos	0.97	0.81	0.88	7459
Probe	0.85	0.68	0.75	2421
R2L	0.80	0.03	0.07	2885
U2R	0.59	0.28	0.38	67
normal	0.66	0.97	0.79	9711
accuracy			0.77	22543
macro avg	0.78	0.56	0.58	22543
weighted avg	0.80	0.77	0.72	22543

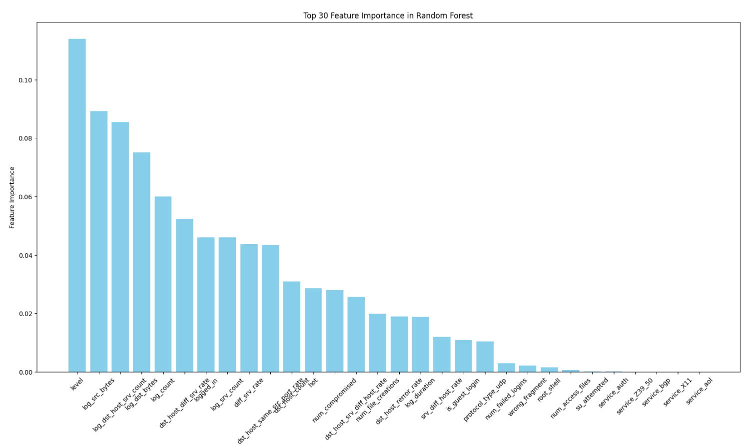
To find the optimal model, random forest classifier was built on Values and the best set of values were used to build the final model.

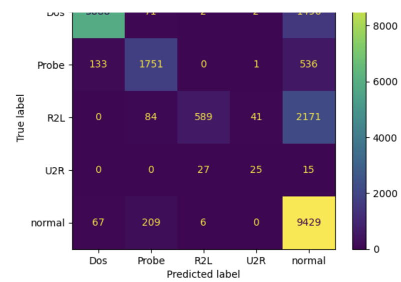
```
Random Forest best n_estimators: 150
Random Forest best max_depth: 11
Random Forest best oob_score: 0.9989159690382937
Random Forest train_score: 0.9991238653871142
Random Forest val_score: 0.9981742409208176
```

```
Accuracy of training data: 0.9991238653871142
Accuracy of testing data: 0.7843676529299561
```

```
report_test=classification_report(y_test, y_pred_test)
print(report_test)
```

	precision	recall	f1-score	support
Dos	0.97	0.79	0.87	7459
Probe	0.83	0.72	0.77	2421
R2L	0.94	0.20	0.34	2885
U2R	0.36	0.37	0.37	67
normal	0.69	0.97	0.81	9711
accuracy			0.78	22543
macro avg	0.76	0.61	0.63	22543
weighted avg	0.83	0.78	0.76	22543





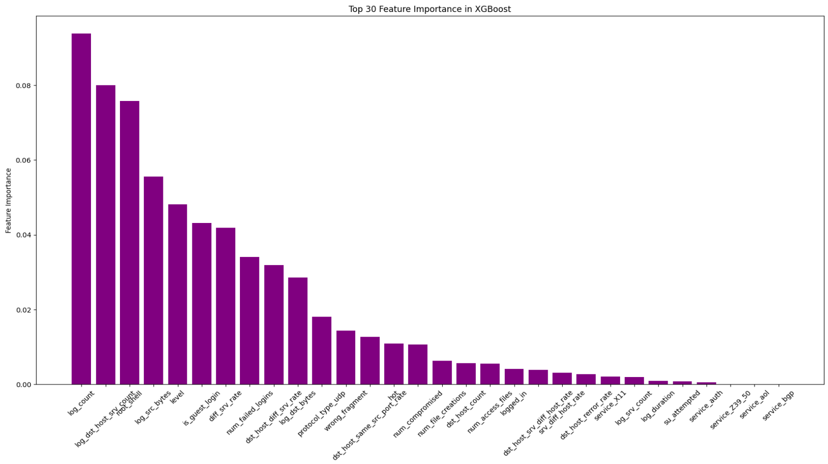
After hypertuning the model there seems to be an increase in the performance in the model. The top 5 features according to the Random Forest classifier are - level, Log_src_bytes, log_dst_host_srv_count, log_dst_bytes, log_counts.

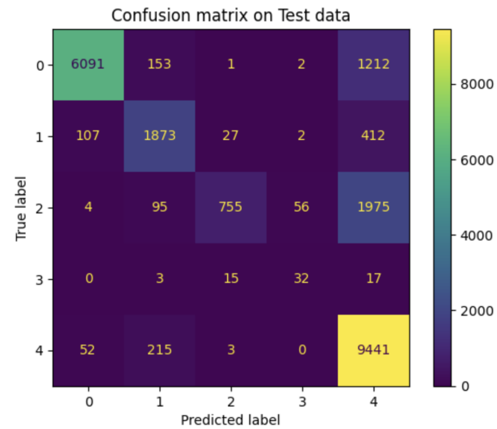
2. XGBoost

To find the optimal model, GridSearch and 3-fold cross-validation is used to find the best 'learning_rate' and 'n_estimators'

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[CV] END ...............learning_rate=0.01, n_estimators=100; total time=  33.2s
[CV] END ...............learning_rate=0.01, n_estimators=100; total time=  28.6s
[CV] END ...............learning_rate=0.01, n_estimators=100; total time=  30.2s
[CV] END ...............learning_rate=0.01, n_estimators=200; total time= 1.0min
[CV] END ...............learning_rate=0.01, n_estimators=200; total time=  57.2s
[CV] END ...............learning_rate=0.01, n_estimators=200; total time=  57.8s
[CV] END ................learning_rate=0.1, n_estimators=100; total time=  32.1s
[CV] END ................learning_rate=0.1, n_estimators=100; total time=  38.1s
[CV] END ................learning_rate=0.1, n_estimators=100; total time=  31.2s
[CV] END ................learning_rate=0.1, n_estimators=200; total time= 1.1min
[CV] END ................learning_rate=0.1, n_estimators=200; total time= 1.0min
[CV] END ................learning_rate=0.1, n_estimators=200; total time= 1.0min
Time taken for model selection (GridSearchCV): 652.58 seconds
Best Hyperparameters: {'learning_rate': 0.1, 'n_estimators': 200}
Best Cross-Validation Score: 0.9997
Train Set Accuracy: 1.0000
Test Set Accuracy: 0.8070
```

	precision	recall	f1-score	support
0	0.97	0.82	0.89	7459
1	0.80	0.77	0.79	2421
2	0.94	0.26	0.41	2885
3	0.35	0.48	0.40	67
4	0.72	0.97	0.83	9711
accuracy			0.81	22543
macro avg	0.76	0.66	0.66	22543
weighted avg	0.84	0.81	0.79	22543





The top 5 features according to XGBoost classifier are - log_count, log_dst_host_srv_count, root_shell, log_src_bytes, level.

3. SVM

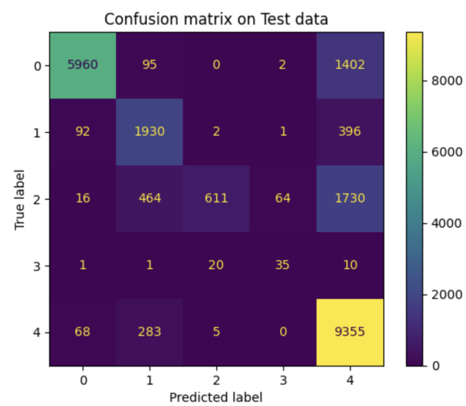
To find the optimal model, GridSearch and 3-fold cross-validation is used to find the best 'kernel'.

```
Fitting 3 folds for each of 3 candidates, totalling 9 fits
[CV] END ...................................kernel=linear; total time= 1.2min
[CV] END ...................................kernel=linear; total time= 1.2min
[CV] END ...................................kernel=linear; total time= 1.3min
[CV] END ......................................kernel=rbf; total time= 1.6min
[CV] END ......................................kernel=rbf; total time= 1.6min
[CV] END ......................................kernel=rbf; total time= 1.7min
[CV] END .....................................kernel=poly; total time= 6.3min
[CV] END .....................................kernel=poly; total time= 6.4min
[CV] END .....................................kernel=poly; total time=11.8min
```

```
Time taken for model selection (GridSearchCV): 2093.20 seconds
Best Hyperparameters: {'kernel': 'rbf'}
Best Cross-Validation Score: 0.9982
Train Set Accuracy: 0.9985
Test Set Accuracy: 0.7936
```

```
[ ] report_test=classification_report(y_test, y_pred)
    print(report_test)
```

```
              precision    recall  f1-score   support

           0       0.97      0.80      0.88      7459
           1       0.70      0.80      0.74      2421
           2       0.96      0.21      0.35      2885
           3       0.34      0.52      0.41        67
           4       0.73      0.96      0.83      9711

    accuracy                           0.79     22543
   macro avg       0.74      0.66      0.64     22543
weighted avg       0.83      0.79      0.77     22543
```

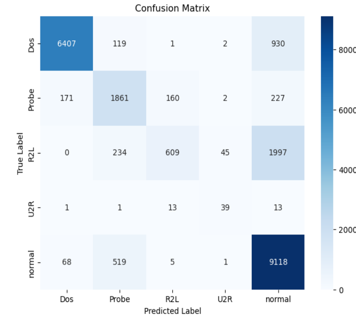


4. Deep Learning model

A simple feedforward neural network was defined using PyTorch, consisting of an input layer (100 neurons), One hidden layer with 64 neurons and a ReLU activation function, An output layer with 5 neurons (one for each class), without a final activation function (as CrossEntropyLoss includes it). The model was trained using the Adam optimizer and the CrossEntropyLoss function in batches of 64. Training was configured to stop automatically if the validation loss did not improve for 5 consecutive epochs (PATIENCE=5).

```
  0%|          | 0/50 [00:00<?, ?it/s]
Epoch [1/50], Train Loss: 0.0178, Train Acc: 99.58%, Val Loss: 0.0162, Val Acc: 99.56%
Validation loss decreased (inf --> 0.0162). Saving model...
Epoch [2/50], Train Loss: 0.0077, Train Acc: 99.81%, Val Loss: 0.0123, Val Acc: 99.65%
Validation loss decreased (0.0162 --> 0.0123). Saving model...
Epoch [3/50], Train Loss: 0.0055, Train Acc: 99.86%, Val Loss: 0.0103, Val Acc: 99.71%
Validation loss decreased (0.0123 --> 0.0103). Saving model...
Epoch [4/50], Train Loss: 0.0046, Train Acc: 99.88%, Val Loss: 0.0074, Val Acc: 99.76%
Validation loss decreased (0.0103 --> 0.0074). Saving model...
Epoch [5/50], Train Loss: 0.0039, Train Acc: 99.90%, Val Loss: 0.0066, Val Acc: 99.77%
Validation loss decreased (0.0074 --> 0.0066). Saving model...
Epoch [6/50], Train Loss: 0.0035, Train Acc: 99.92%, Val Loss: 0.0068, Val Acc: 99.82%
Validation loss did not improve for 1 epoch(s).
Epoch [7/50], Train Loss: 0.0030, Train Acc: 99.93%, Val Loss: 0.0050, Val Acc: 99.88%
Validation loss decreased (0.0066 --> 0.0050). Saving model...
Epoch [8/50], Train Loss: 0.0026, Train Acc: 99.94%, Val Loss: 0.0066, Val Acc: 99.83%
Validation loss did not improve for 1 epoch(s).
Epoch [9/50], Train Loss: 0.0024, Train Acc: 99.95%, Val Loss: 0.0045, Val Acc: 99.89%
Validation loss decreased (0.0050 --> 0.0045). Saving model...
Epoch [10/50], Train Loss: 0.0022, Train Acc: 99.95%, Val Loss: 0.0047, Val Acc: 99.90%
Validation loss did not improve for 1 epoch(s).
Epoch [11/50], Train Loss: 0.0020, Train Acc: 99.96%, Val Loss: 0.0062, Val Acc: 99.86%
Validation loss did not improve for 2 epoch(s).
Epoch [12/50], Train Loss: 0.0017, Train Acc: 99.96%, Val Loss: 0.0045, Val Acc: 99.89%
Validation loss did not improve for 3 epoch(s).
Epoch [13/50], Train Loss: 0.0017, Train Acc: 99.96%, Val Loss: 0.0051, Val Acc: 99.88%
Validation loss did not improve for 4 epoch(s).
Epoch [14/50], Train Loss: 0.0015, Train Acc: 99.97%, Val Loss: 0.0045, Val Acc: 99.88%
Validation loss did not improve for 5 epoch(s).

Early stopping triggered after 14 epochs.

Training finished in 173.79 seconds.
--------------------
```

The best validation performance (lowest loss of 0.0045, highest accuracy of 99.89%) was achieved at Epoch 9 and Early stopping correctly triggered at Epoch 14.



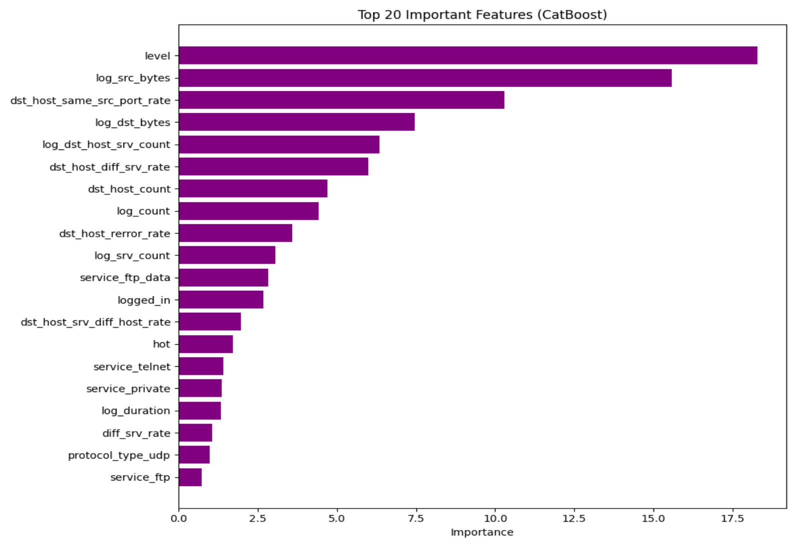

```
Classification Report:
              precision    recall  f1-score   support

         Dos       0.96      0.86      0.91      7459
       Probe       0.68      0.77      0.72      2421
         R2L       0.77      0.21      0.33      2885
         U2R       0.44      0.58      0.50        67
      normal       0.74      0.94      0.83      9711

    accuracy                           0.80     22543
   macro avg       0.72      0.67      0.66     22543
weighted avg       0.81      0.80      0.78     22543
```

5. CatBoost

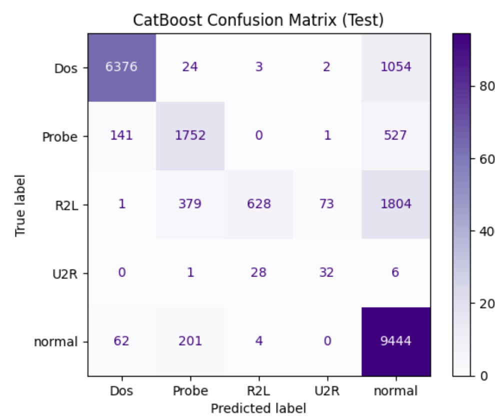
To find the optimal model, GridSearch and 3-fold cross-validation i

```
Best CatBoost Parameters: {'learning_rate': 0.2}
```

depth'.




```
Best Hyperparameters: {'learning_rate': 0.2}
Best Cross-Validation Score: 0.9997
Train Set Accuracy: 1.0000
Test Set Accuracy: 0.8088
Classification Report (Test Set):
              precision    recall  f1-score   support

         Dos       0.97      0.85      0.91      7459
       Probe       0.74      0.72      0.73      2421
         R2L       0.95      0.22      0.35      2885
         U2R       0.30      0.48      0.37        67
      normal       0.74      0.97      0.84      9711

    accuracy                           0.81     22543
   macro avg       0.74      0.65      0.64     22543
weighted avg       0.84      0.81      0.79     22543
```



The top 5 features according to XGBoost classifier are - level, log_src_bytes, dst_host_same_src_port_rate, log_dst_bytes, and log_dst_host_srv_count.

Summary

Models	Test Accuracy	Precision	Recall	F1-score	Remarks
Random Forest	78.43	83	78	76	Best n_estimator=150, max_depth=11
XGBoost	80.7	84	81	79	Best n_estimator=200, learning_rate=0.1
SVM	79.36	83	79	77	Best kernel=rbf

Deep Learning neural network	80	81	80	78	Reached optimum at Epoch [9/50] Train Loss: 0.0015, Val Loss: 0.0045
CatBoost	80.88	84	81	79	Best learning rate=0.2

From the above table, the hyper-tuned CatBoost and XGBoost models seem to be working the best followed by the Deep Learning model, SVM with ‘rbf’ kernel and Random Forest model. From the confusion matrices in the above section we can observe that all the models except random forest (for U2R) have predicted the majority of samples in each class correctly.
Furthermore, from the feature importances graphs produced by random forest, xgboost, and catboost models we can see that they have a few common features like - level, Log_src_bytes, log_dst_host_srv_count, log_dst_bytes, log_counts, etc.. as the top 20 important features.