INFO7390: Advances Data Sci/Architecture

Project 4: Anomaly Detection

Naman Gupta
(NUID: 002729751)
Northeastern University
Boston Campus
gupta.naman@northeastern.edu

Introduction

The goal of this project is to develop a model for classifying emails as spam or non-spam based on various features. The dataset, obtained from the UC Irvine Machine Learning Repository, includes 48 continuous real attributes representing word and character frequencies, as well as features related to the length of sequences of consecutive capital letters. The last column of the dataset denotes whether the email is considered spam (1) or not (0).

The task involves detecting unsolicited commercial emails (spam) using machine learning models. Anomaly Detection: Since spam emails are considered anomalies in this context, the project utilizes anomaly detection techniques.

In the previous assignment, we tried with baseline models which didn't give impressive results. Hence, this will be a continuation of the previous assignment on trying a neural network based model for anomaly detection.

Source: http://archive.ics.uci.edu/dataset/94/spambase

Data Exploration and Handling

A. Data Profiling & Exploration

The dataset is explored using the ydata-profiling library to generate a profile report. Observations include left-skewed distributions in word and character frequencies, imbalanced class distribution (60.6% non-spam, 39.4% spam), and the need for careful consideration of evaluation metrics due to class imbalance.

Dataset statistics

Number of variables	58
Number of observations	4601
Missing cells	0
Missing cells (%)	0.0%
Total size in memory	2.0 MiB
Average record size in memory	464.0 B

Variable types

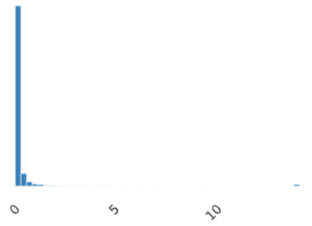
Numeric	58

Figure: 1 – Dataset statistics

word_freq_address
Real number (ℝ)

ZEROS

Distinct	171	**Minimum**	0
Distinct (%)	3.7%	**Maximum**	14.28
Missing	0	**Zeros**	3703
Missing (%)	0.0%	**Zeros (%)**	80.5%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	0.2130145621	**Memory size**	36.1 KiB

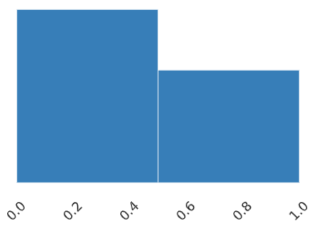


More details

class
Real number (ℝ)

ZEROS

Distinct	2	**Minimum**	0
Distinct (%)	< 0.1%	**Maximum**	1
Missing	0	**Zeros**	2788
Missing (%)	0.0%	**Zeros (%)**	60.6%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	0.3940447729	**Memory size**	36.1 KiB



More details

Figure: 2 - Dataset exploration

B. Model Selection and Development

1. **Isolation Forest Model**:

- The dataset is split into training, validation, and test sets.
- The Isolation Forest model is trained on the training set and evaluated on the validation set.
- Performance metrics include classification accuracy, precision, recall, F1-score, and confusion matrix.

2. **One-Class SVM Model**:

- Similar steps as the Isolation Forest model are followed.
- Performance metrics are assessed on the validation set.

3. **Random Forest Model**:

- A Random Forest Classifier is trained and evaluated on the validation set.

- Robust performance is observed with high accuracy (96%) and balanced precision, recall, and F1 scores.

4. Autoencoders:

- Autoencoder is a type of neural network used for unsupervised learning and dimensionality reduction. In the context of anomaly detection, autoencoders learn to reconstruct input data, aiming to capture normal patterns while highlighting anomalies.
- The encoder compresses input data into a lower-dimensional representation, and the decoder reconstructs the original input from this representation. Anomalies may result in higher reconstruction errors, making autoencoders effective for identifying deviations from normal patterns.
- Interpretation involves analyzing precision, recall, and AUC-PR to strike a balance between correctly identifying anomalies and minimizing false positives.

Model Evaluation and Challenges

A. Isolation Forest and One-Class SVM

- Both models assume anomalies are rare, impacting performance due to the significant class imbalance in the dataset.
- Accuracy is around 40-50%, emphasizing the need for improvement.
- Challenges in correctly identifying both spam and non-spam instances are evident in the confusion matrices.

```
IsolationForest
IsolationForest(contamination=0.4, random_state=42)

Validation Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.65      0.64       399
           1       0.50      0.48      0.49       291

    accuracy                           0.58       690
   macro avg       0.56      0.56      0.56       690
weighted avg       0.57      0.58      0.58       690

Validation Accuracy Score: 0.58
Validation Confusion Matrix:
[[258 141]
 [151 140]]
```

Figure: 3 – Isolation Forest Performance on Validation data

```
OneClassSVM
OneClassSVM(nu=0.4)

Validation Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.61      0.59       399
           1       0.40      0.35      0.37       291

    accuracy                           0.50       690
   macro avg       0.48      0.48      0.48       690
weighted avg       0.49      0.50      0.50       690

Validation Accuracy Score: 0.5
Validation Confusion Matrix:
[[245 154]
 [189 102]]
```

Figure: 4 – One Class SVM Performance on Validation data

```
RandomForestClassifier
RandomForestClassifier(class_weight='balanced', random_state=42)
```

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

           0       0.96      0.97      0.96       399
           1       0.95      0.94      0.95       291

    accuracy                           0.96       690
   macro avg       0.95      0.95      0.95       690
weighted avg       0.96      0.96      0.96       690

Validation Accuracy Score: 0.96
Validation Confusion Matrix:
[[386  13]
 [ 18 273]]
```

Figure: 5 – Random Forest Performance on Validation data

B. Transformation of Dataset

- To address the class imbalance issue, the dataset is transformed to make spam instances rare (5%).
- Rows corresponding to spam are reduced, resulting in a dataset with ~5% spam and ~95% non-spam.

```
Transfromed dataframe shape:
 (2933, 58)

Original dataframe class distribution:
 0    60.595523
1    39.404477
Name: class, dtype: float64

Transfromed dataframe class distribution:
 0    95.056256
1     4.943744
Name: class, dtype: float64
```

Figure: 6 -Transformed data statistics

C. Isolation Forest and One-Class SVM on Transformed Dataset

- Models are re-evaluated on the transformed dataset.
- Improved accuracy (92%) is observed for the Isolation Forest model, but imbalanced performance remains.
- Similar findings are observed for the One-Class SVM model.

```
IsolationForest
IsolationForest(contamination=0.05, random_state=42)
```

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

           0       0.95      0.97      0.96       414
           1       0.22      0.15      0.18        26

    accuracy                           0.92       440
   macro avg       0.59      0.56      0.57       440
weighted avg       0.90      0.92      0.91       440

Validation Accuracy Score: 0.92
Validation Confusion Matrix:
[[400  14]
 [ 22   4]]
```

Figure: 7 – Isolation Forest Performance on Validation Data After Transformation

```
OneClassSVM
OneClassSVM(nu=0.05)
```

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

           0       0.95      0.97      0.96       414
           1       0.25      0.15      0.19        26

    accuracy                           0.92       440
   macro avg       0.60      0.56      0.57       440
weighted avg       0.91      0.92      0.91       440

Validation Accuracy Score: 0.92
Validation Confusion Matrix:
[[402  12]
 [ 22   4]]
```

Figure: 8 – One Class SVM Performance on Validation Data After Transformation

D. Hyperparameter Tuning for Isolation Forest

- GridSearchCV is employed for hyperparameter tuning.
- The best model achieves an accuracy of 93%, with improved precision in identifying non-spam emails.

```
Best Parameters: {'contamination': 0.01, 'max_samples': 1.0, 'n_estimators': 50}

Validation Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.97       414
           1       0.29      0.08      0.12        26

    accuracy                           0.93       440
   macro avg       0.62      0.53      0.54       440
weighted avg       0.91      0.93      0.92       440

Validation Accuracy Score: 0.93
Validation Confusion Matrix:
[[409   5]
 [ 24   2]]
```

Figure: 9 – Isolation Forest Performance on Validation Data After Transformation
With Hyperparameter Tuning

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

           0       0.96      0.99      0.97       420
           1       0.17      0.05      0.08        20

    accuracy                           0.95       440
   macro avg       0.56      0.52      0.52       440
weighted avg       0.92      0.95      0.93       440

Test Accuracy Score: 0.95
Test Confusion Matrix:
[[415   5]
 [ 19   1]]
```

Figure: 10 – Isolation Forest Performance on Test Data After Transformation
With Hyperparameter Tuning

E. Interpreting the Best Model Results and Identifying the Next Steps

- We saw that the hyperparameter-tuned Isolation Forest Model could achieve a good accuracy score on the transformed dataset. However, the precision, recall, and f1-score for the minority class (spam emails) remains too low, which means that the model is still quite inefficient in detecting spam emails.

- This can also be understood from the confidence matrix. Since the class distribution in the transformed dataset contains a huge imbalance (95:5), one cannot trust the accuracy score to evaluate the model.
- On studying the correlation of features with the target variable, we found that a lot of features in the data do not contribute to building the decision of the target class. To proceed further in feature selection, we used a Random Forest Classifier to identify the most impactful features for building the decision of the target class.
- Also, to improve the precision in identifying spam emails, we introduced another model "Autoencoder", which brings complexity of neural networks.

```
[ ]
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Standardize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train[selected_features])
X_val_scaled = scaler.fit_transform(X_val[selected_features])
X_test_scaled = scaler.fit_transform(X_test[selected_features])

# Build the autoencoder model
input_dim = X[selected_features].shape[1]
encoding_dim = 32
hidden_dim = 16

autoencoder = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(input_dim,)),
    tf.keras.layers.Dense(encoding_dim, activation='relu'),
    tf.keras.layers.Dense(hidden_dim, activation='relu'),
    tf.keras.layers.Dense(encoding_dim, activation='relu'),
    tf.keras.layers.Dense(input_dim, activation='sigmoid')
])

# Mean Squared Error (MSE) as loss function
autoencoder.compile(optimizer='adam', loss='mse')
```

Figure: 11 – Multilayered Autoencoder Structure

F. Autoencoder on Transformed Dataset

- Autoencoder achieved higher precision for anomalies (0.22) compared to initial results (Isolation Forest - 0.17), indicating a reduction in false positives.
- Autoencoder demonstrated a significant improvement in recall for anomalies (0.65), a substantial increase from the initial result (Isolation Forest - 0.05). This suggests a better ability to capture true anomalies.
- The overall accuracy dropped to 0.88, and the weighted F1-score decreased to 0.91, indicating a trade-off between precision and recall. This trade-off highlights the nuanced anomaly detection capability of the autoencoder.
- Autoencoder's introduction of complexity, as a neural network, resulted in a more nuanced anomaly detection capability, showcasing improvements in recall while managing precision.

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier

# Example using RandomForest for feature importance
model = RandomForestClassifier()
selector = SelectFromModel(model)
selector.fit(X, y)

# Get selected features
selected_features = X.columns[selector.get_support()]
```

```
print(selected_features)
```

```
Index(['word_freq_all', 'word_freq_our', 'word_freq_over', 'word_freq_remove',
       'word_freq_will', 'word_freq_free', 'word_freq_business',
       'word_freq_you', 'word_freq_credit', 'word_freq_your', 'word_freq_000',
       'word_freq_money', 'char_freq_(', 'char_freq_!', 'char_freq_$',
       'capital_run_length_average', 'capital_run_length_longest',
       'capital_run_length_total'],
      dtype='object')
```

Figure: 12 – Feature selection using Random Forest Regressor

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

           0       0.96      0.99      0.97       414
           1       0.57      0.31      0.40        26

    accuracy                           0.95       440
   macro avg       0.76      0.65      0.69       440
weighted avg       0.93      0.95      0.94       440

Validation Accuracy Score: 0.95
Validation Confusion Matrix:
[[408   6]
 [ 18   8]]
```

Figure: 13 – Autoencoder Performance on Validation Data

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

           0       0.98      0.89      0.94       420
           1       0.22      0.65      0.33        20

    accuracy                           0.88       440
   macro avg       0.60      0.77      0.63       440
weighted avg       0.95      0.88      0.91       440

Test Accuracy Score: 0.88
Test Confusion Matrix:
[[375  45]
 [  7  13]]
```

Figure: 14 – Autoencoder Performance on Test Data

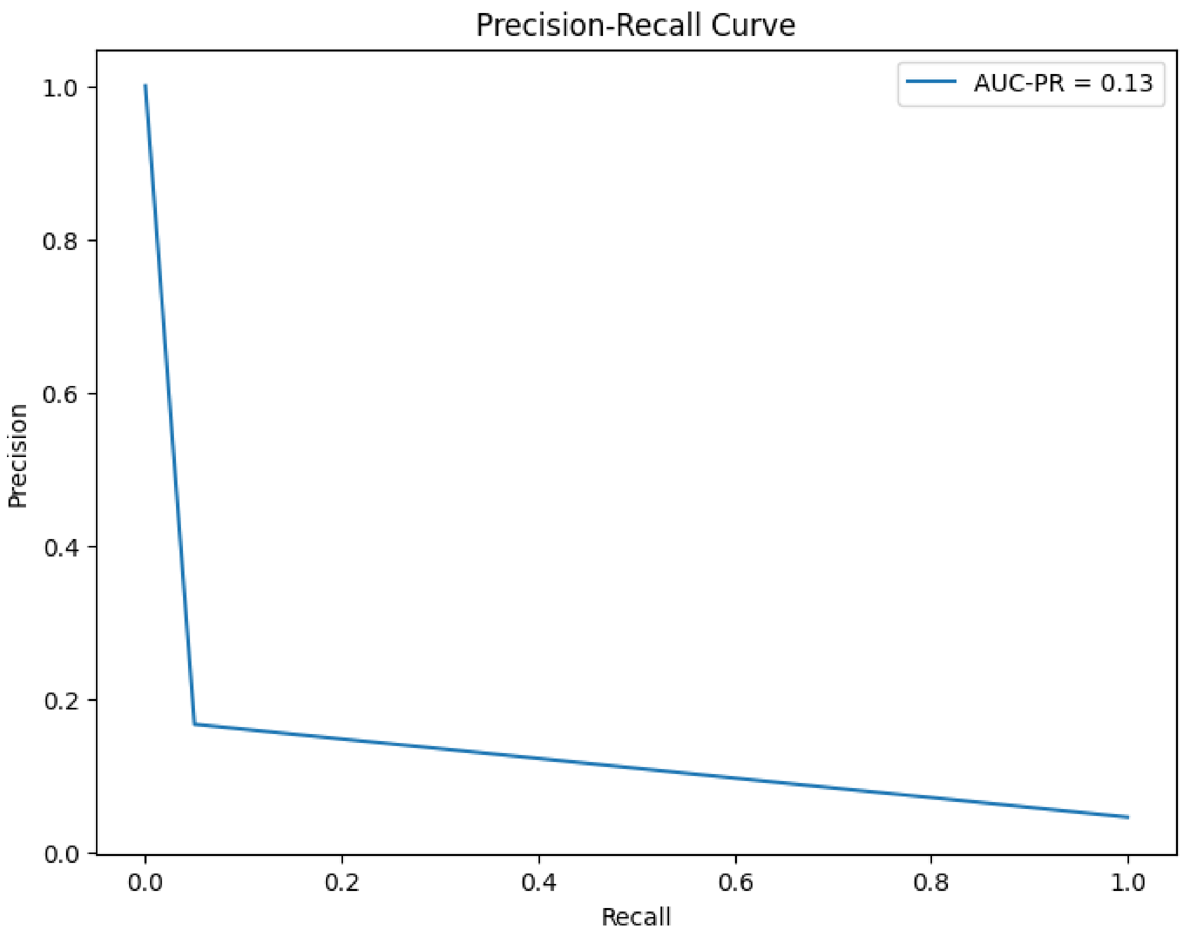


Figure: 15 – Precision-Recall Curve for Isolation Forest (Hyperparameter Tuned)

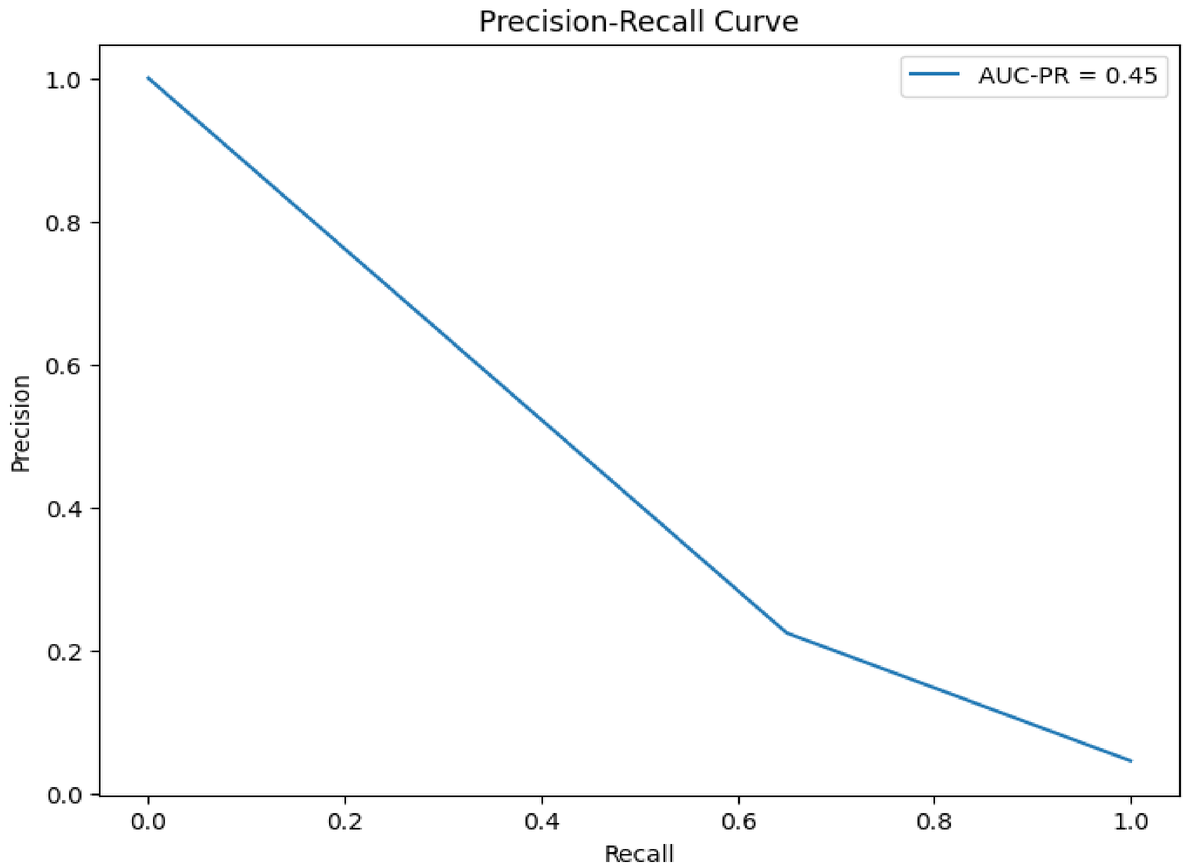


Figure: 16 – Precision-Recall Curve for Autoencoders

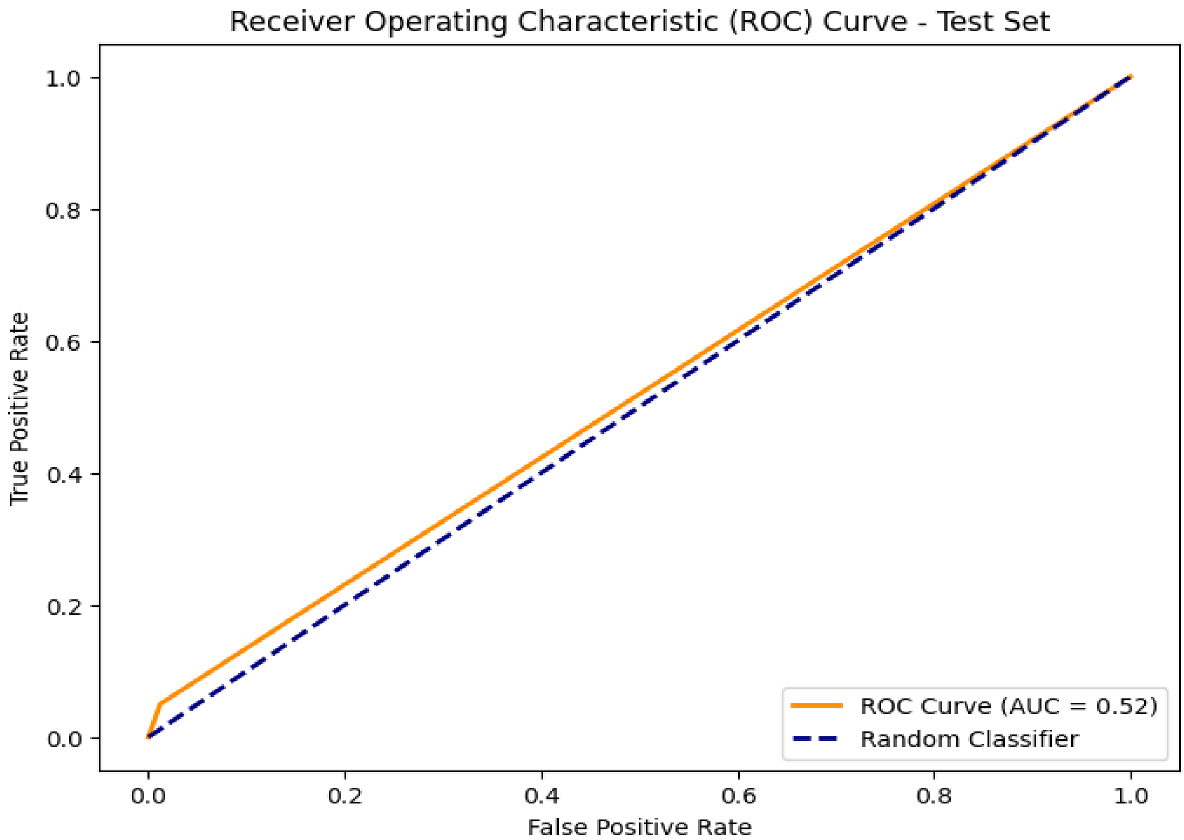


Figure: 17 – ROC Curve for Isolation Forest (Hyperparameter Tuned)

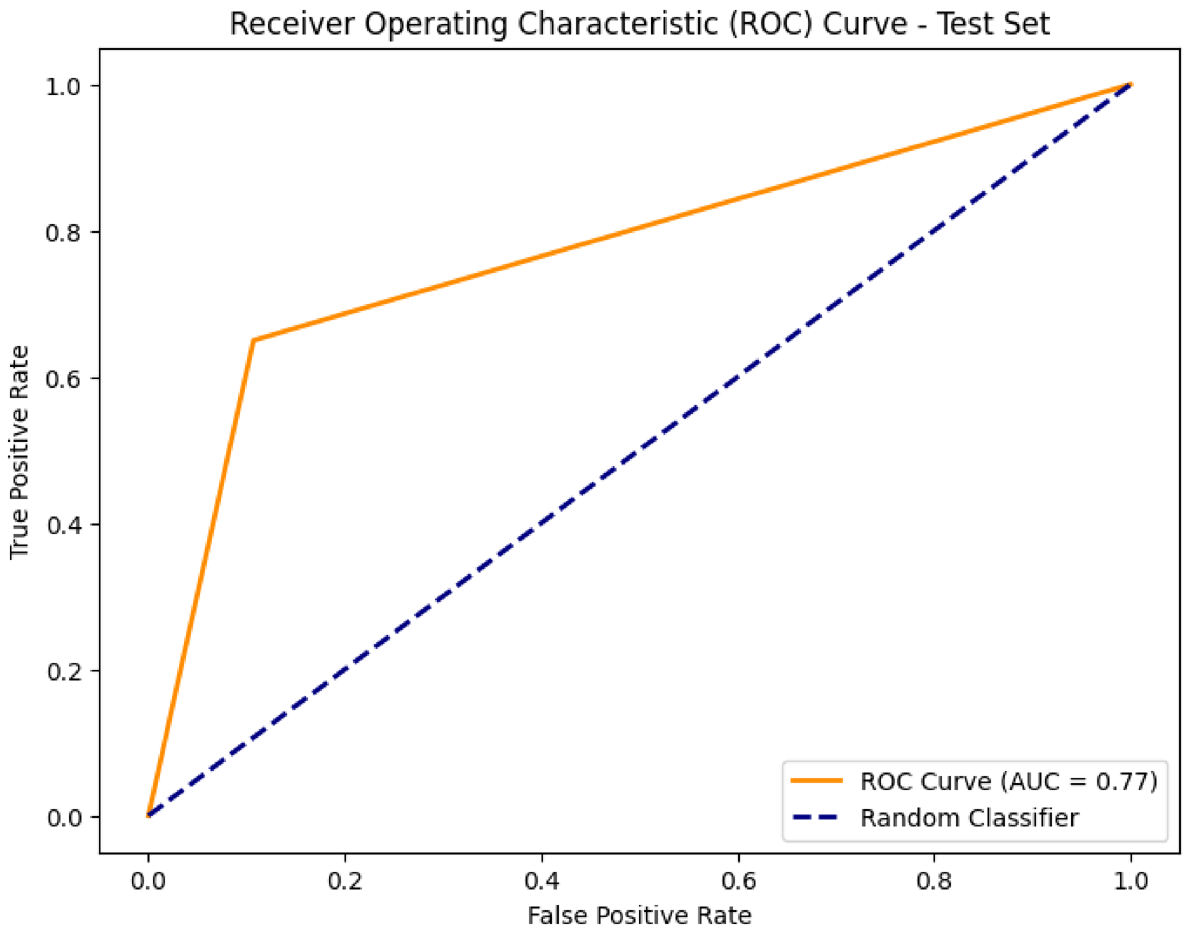


Figure: 18 – ROC Curve for Autoencoders

Deployment

The best-trained model out of the above experiments model was saved using the "joblib" Python library. This model has been utilized in recognizing spam for other datasets with the same structure. A Streamlit application has been built to test this model.

The application has features as follows:

- Uploading a dataset in CSV format
- Predicting spam emails using the best model
- Visualizing the model's performance on the dataset
- Generating and visualizing profiling report of the uploaded dataset

Naman Gupta
002729751

Spam Detection App

Choose a CSV file for testing

Drag and drop file here
Limit 200MB per file • CSV

Browse files

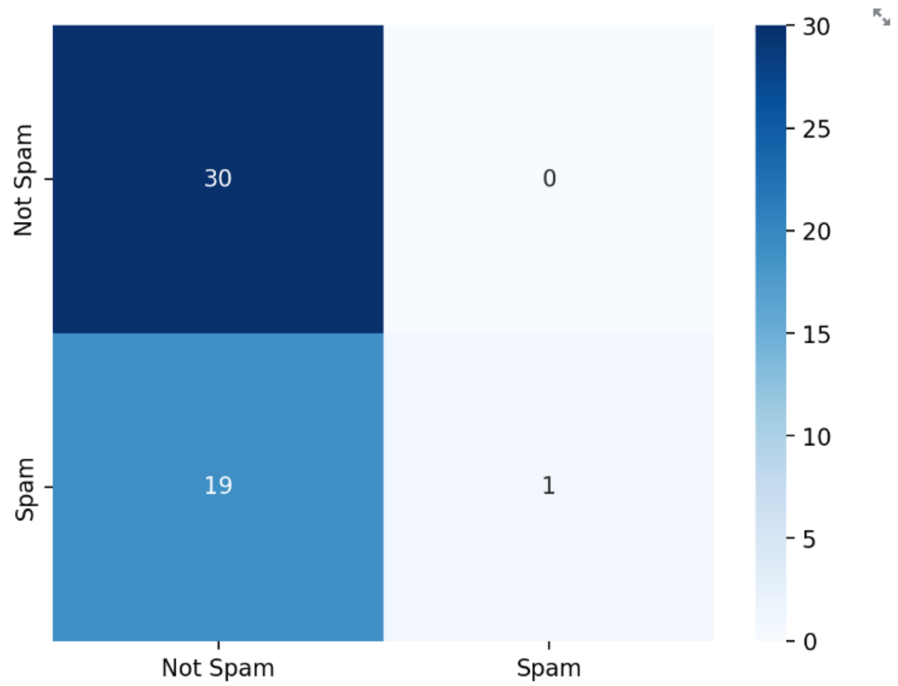
spam_test.csv 11.7KB

Uploaded Test Data:

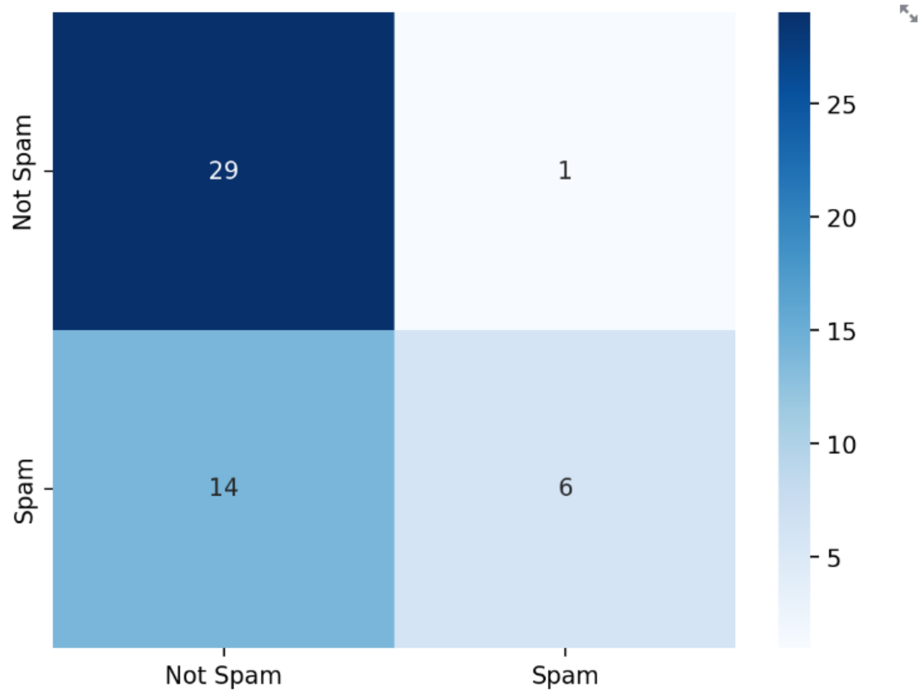
	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	2.17	0	0	0
1	0.27	0	0.27	0	0	0	0	0	0	0	0	0.81	0
2	0.86	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0.28	0.28	0	0.86	0	0	0	0	0.28	0.28	0	0
4	0.26	0	0.53	0	0	0.26	0	0	0	0	0	1.06	0
5	0	0	0	0	0	0	2.3	0	0	0	0.76	2.3	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0.32	0.09	0.6	0	2.04	0.13	0	0	0.09	0.69	0.32	0.79	0.27
9	0	0	0	0	0	0	0	1.29	0	0.43	0	0	0

Detect Spams

Confusion Matrix using Hyperparameter tuned Isolation Forest Model:



Confusion Matrix using Multi-layered Autoencoder Model:



Data Exploration Report

Profiling Report

Overview

Overview Alerts 58 Reproduction

Dataset statistics

Number of variables	58
Number of observations	50
Missing cells	0
Missing cells (%)	0.0%
Total size in memory	22.8 KiB
Average record size in memory	466.6 B

Variable types

Numeric	58

Conclusion & Learning

In conclusion, the evaluation of anomaly detection models, particularly the Isolation Forest and Autoencoder, revealed valuable insights into their performance on an imbalanced dataset with a class distribution of 95:5. The Isolation Forest, despite achieving a high overall accuracy and respectable weighted F1-score, faced challenges in precision and recall for identifying anomalies. On the other hand, the Autoencoder exhibited significant improvements in the detection of anomalies, showcasing higher precision and a substantial increase in recall.

The trade-off between precision and recall in the Autoencoder, leading to a lower overall accuracy and weighted F1-score, underscores the model's nuanced anomaly detection capability. Sensitivity to the imbalanced class distribution highlights the importance of carefully handling such scenarios and fine-tuning decision thresholds to optimize the desired trade-off.

The iterative nature of the analysis suggests further exploration, including hyperparameter adjustments, alternative models, and strategies to address class imbalance. These findings contribute to a deeper understanding of the strengths and challenges posed by different anomaly detection techniques in the specific context of the dataset. The insights gained pave the way for refining models, enhancing anomaly identification, and ultimately improving the overall effectiveness of anomaly detection in the given data.

References

1. Spambase - UCI Machine Learning Repository
2. Welcome - YData Profiling
3. sklearn.ensemble.IsolationForest — scikit-learn 1.3.2 documentation
4. sklearn.svm.OneClassSVM — scikit-learn 1.3.2 documentation
5. sklearn.ensemble.RandomForestClassifier — scikit-learn 1.3.2 documentation
6. Intro to Autoencoders | TensorFlow Core
7. Streamlit documentation
8. ChatGPT (openai.com)