Sophea Bonne, 352901
Joshua Cohen-Dumani, 311105

ML for Predictive Maintenance Applications: Assignment 2

Since our methodologies for both the pump and the valve are very identical, we will explain our methodology only with the part on the Pump, and for the valve we will simply show our results (plots, metrics), and our interpretations.

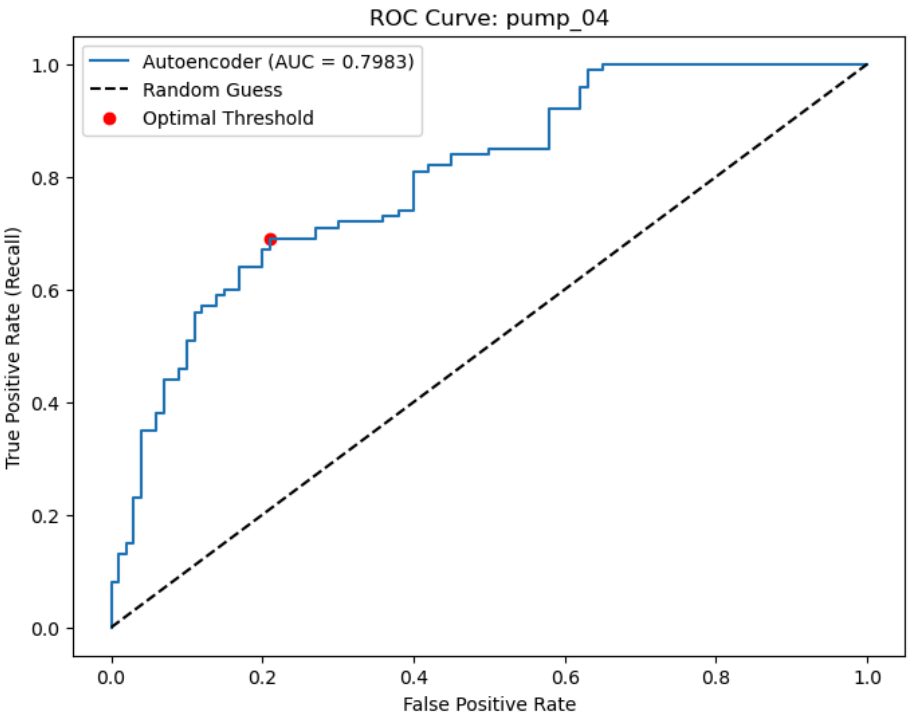
PUMP

Question 1

Given the provided code, fill in the blank spaces to extract signal features, train a simple AutoEncoder that reconstructs the inputs MEL-Spectogram, and report the AUC score.

In this report we will go over the methodology used for anomaly detection in the pump and valve datasets. As defined in the boilerplate code, we will be using the mel spectrogram as input features. The train dataset contains only healthy data, while the test dataset includes both normal and anomalous data. The test dataset for the pump is balanced, with 100 normal and 100 anomalous samples.

For the first question, we trained a symmetric dense 3-layer AutoEncoder. We decided to train for 10 epochs, and have a batch size of 32. As suggested by the boilerplate code we used the Adam Optimizer and Mean Squared Error Loss. For this question we were simply asked to provide the AUC, which we have computed as 0.7983, which is pretty good (much better than last week) but can probably still be improved further.



In the adjacent figure, we can see the ROC curve of the simple AutoEncoder, which we will revisit in more detail for question 4. The red dot is our optimal threshold, which was determined using the Youden J statistic (again, will be further explored in Q4)

Question 2

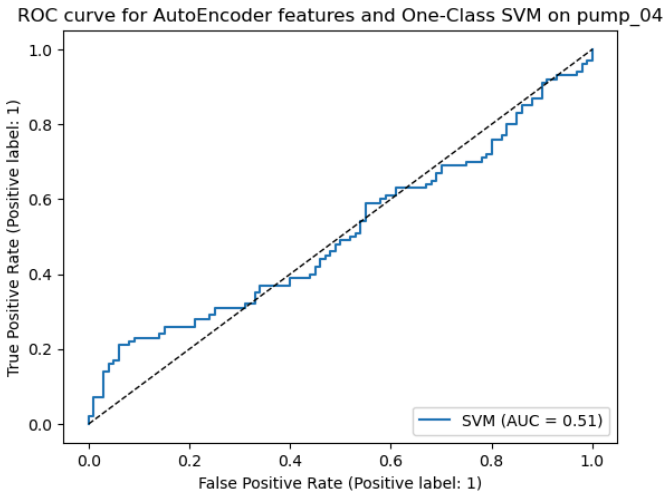
From the trained AutoEncoder, use the bottleneck features to train a (1) One-Class SVM and (2) Isolation Forest, and report the AUC.

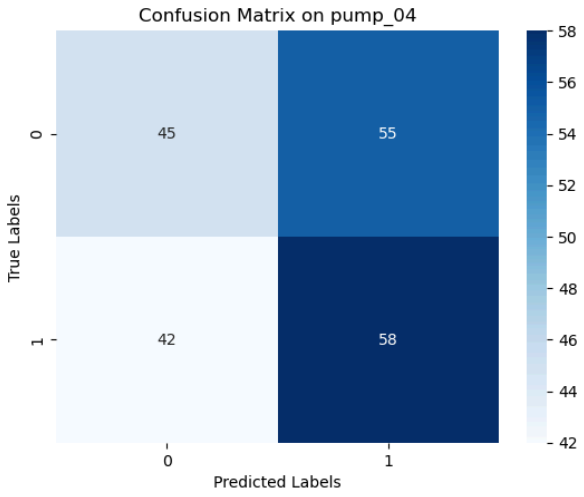
<u>One-Class SVM</u>

For this question, we first trained a One-Class SVM, using the features extracted by the trained AutoEncoder model described above. This simple model actually performed quite poorly, not really doing better than random, with an AUC score of only 0.5097. We used OneClassSVM().decision_function(features) to compute the ROC curves. For reference, the SVM had the following evaluation metrics:

- AUC: 0.5097
- Accuracy: 0.5150
- True Positive Rate (Recall): 0.5800
- False Positive Rate: 0.5500
- Precision: 0.5133
- F1-Score: 0.5446

One interpretation of this poor performance could be that the autoencoder features are not





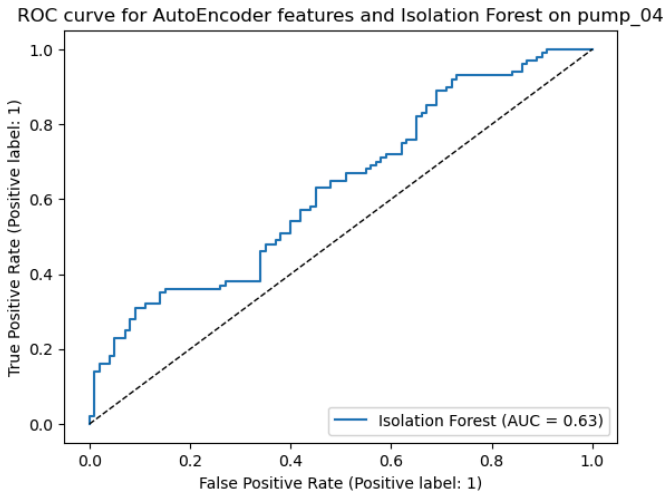
very helpful for the model, which might interpret them badly. For example, since the autoencoder's features are non-linear by nature (versus PCA), the model might have trouble using the information in the features effectively.

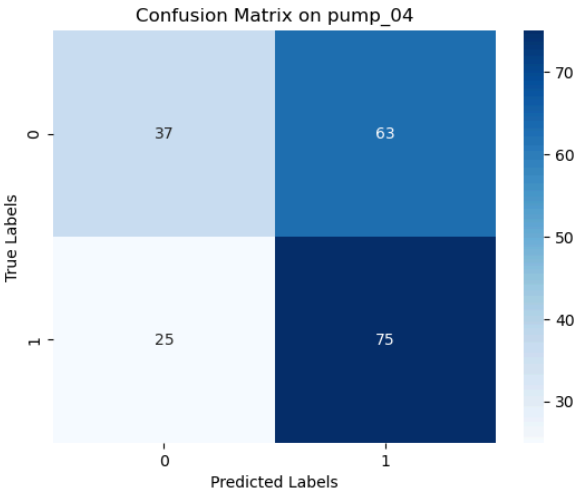
<u>Isolation Forest</u>

Then, we compared the One-Class SVM with an Isolation Forest, which are generally more robust to more complex and non-linear inputs than an SVM-based model, using the same extracted features from the autoencoder model described above. While this model outperformed the One-Class SVM, getting slightly better than random results, this model performed poorly, with an AUC of 0.6296. Below are the evaluation metrics:

- AUC: 0.6296
- Accuracy: 0.5600
- True Positive Rate (Recall): 0.7500
- False Positive Rate: 0.6300
- Precision: 0.5435
- F1-Score: 0.6303

While the results are still not great, they are slightly better, which does suggest that the isolation forest might be working better with the complex autoencoder features.





Question 3

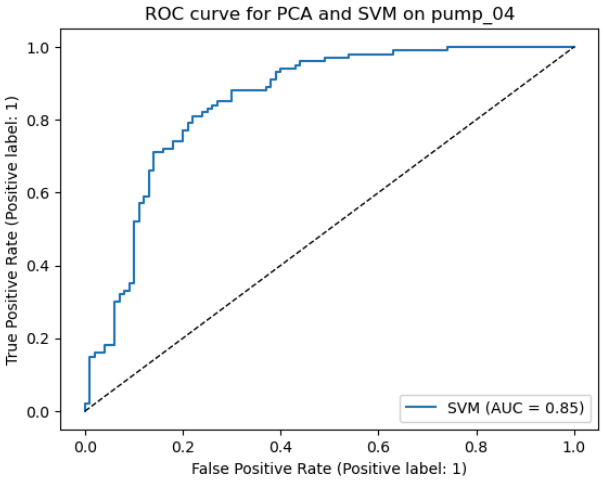
Instead of using the AutoEncoder feature, project the MEL-spectrogram into smaller dimensions using a PCA to train a (1) One-Class SVM and (2) Isolation Forest, and report the AUC.

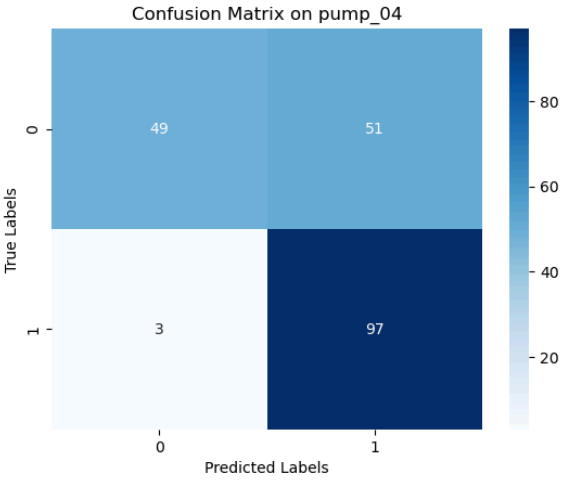
For this question, instead of using AutoEncoder features to train the One-Class SVM and Isolation Forest, we will now try to train them using the PCA approach. This method, which is simpler and lower-dimensional than the AutoEncoder, might work better for us. In this question we used PCA with 64 components, as suggested by the boilerplate code.

<u>One-Class SVM</u>

The One-Class SVM, trained on PCA features, actually performed much better than our previous models. Indeed, our model has an AUC of 0.8507, significantly better than random. For reference, our evaluation metrics are:

- AUC: 0.8507
- Accuracy: 0.7300
- True Positive Rate (Recall): 0.9700
- False Positive Rate: 0.5100
- Precision: 0.6554





- F1-Score: 0.7823

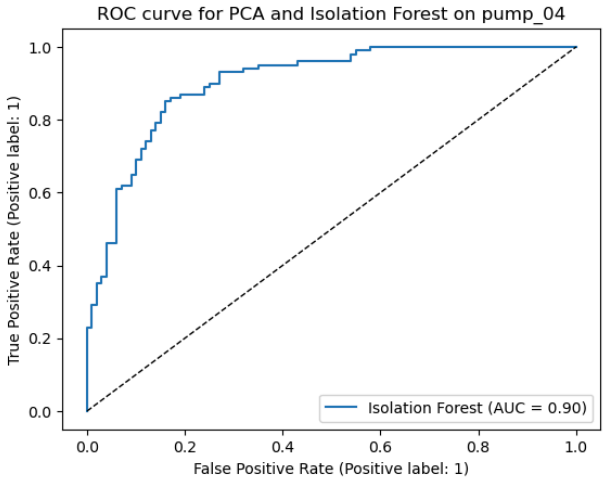
As expected, the simpler and more linear features of PCA align better with SVM's algorithm, which tries to find a hyperplane which encapsulates "normal data" effectively. Since there are more features with PCA (64 vs the 32 in the AutoEncoder) and the features are perhaps more "compatible" with these algorithms, it is not so surprising that our model is much better when using PCA features.

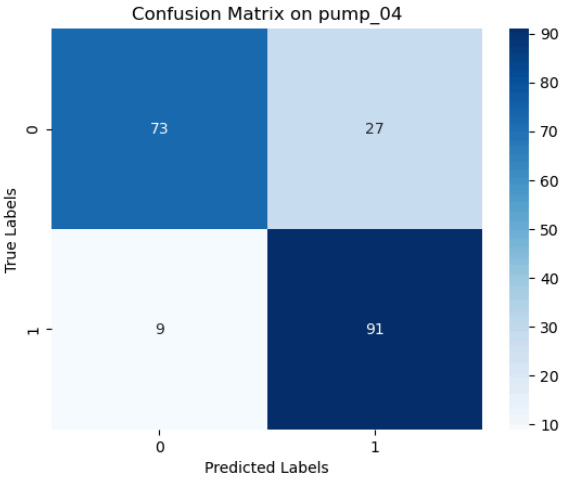
Isolation Forest

Similarly, the Isolation Forest using PCA also massively outperformed the models using AutoEncoder features, with an AUC of 0.9038. For reference, our evaluation metrics are:

- AUC Score: 0.9038
- Accuracy: 0.8200
- True Positive Rate (Recall): 0.9100
- False Positive Rate: 0.2700
- Precision: 0.7712
- F1-Score: 0.8349

This result is quite good, since this model can quite reliably find anomalies while minimizing false positives. This could be due to the nature of the pump data. As seen in exercise 1, the anomalies were not always homogeneous in *how* they are anomalous, which for SVM, which somehow make some assumptions as to the distribution of the data, might pose an issue. Isolation forests on the other hand are independent of the underlying distribution, which might explain why it is more effective in this scenario. Another reason could simply be that isolation forests are less sensitive to hyperparameters, which could explain the difference since we did not have time to grid-search the best params for all the models.



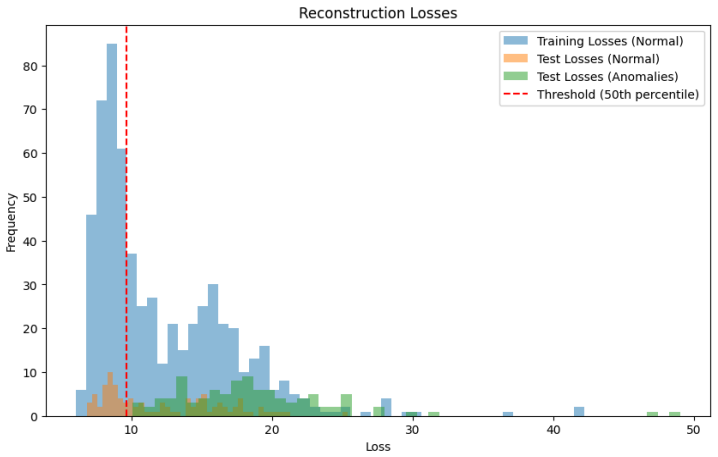


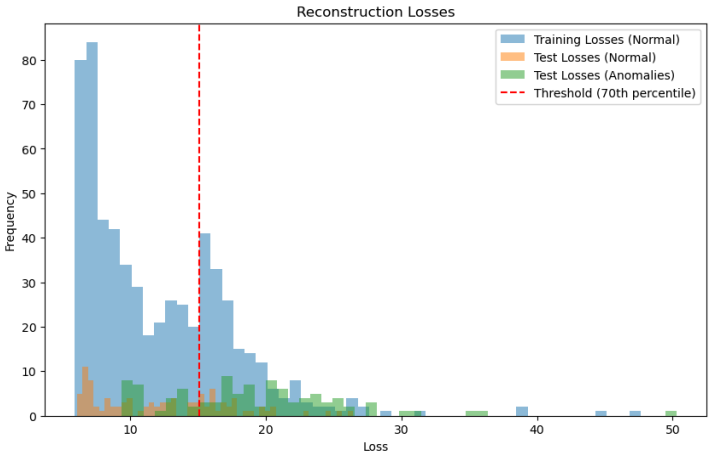
Question 4

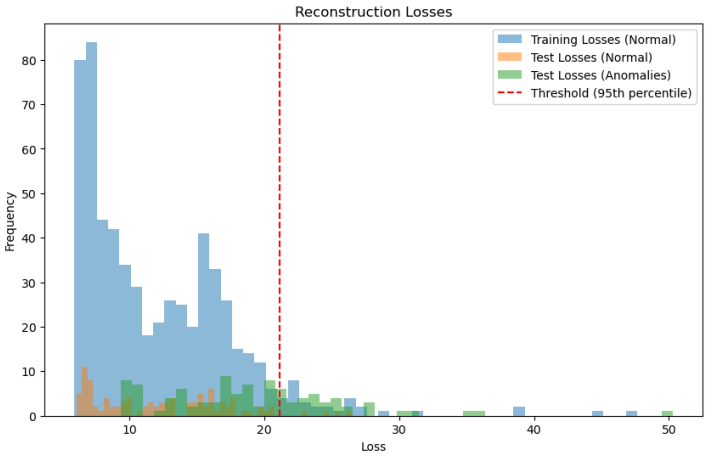
Find a suitable threshold for separating the anomalies from the AutoEncoder reconstruction score, to compute the accuracy, True Positive Rate (TPR), False Positive Rate (FPR), and F1-score.

Manual Thresholding

In order to find a “good” threshold for our AutoEncoder, we will be using the reconstruction losses. Our first approach was to run the training data through our trained model (ie. by using the training data on the test loop), and collecting the losses. This then allowed us to compute different thresholds based on percentile. Below we see the plots experimenting with thresholds of 50th percentile, 75th percentile, and 95th percentile. The evaluation metrics at each threshold are documented below the plots. As expected, as the threshold increases we move to a more restrictive model, that thus has less false positives. It was hard for us to build conviction using this approach (the best F1 score is for the threshold at the 50th percentile but this seemed quite arbitrary and sub-optimal). This has led to us trying to find other ways to find the optimal threshold. We will outline 2 more methods.







Evaluation Metrics - Manual Thresholding

Threshold (at 50th percentile): 10.564491271972656
Accuracy: 0.6550
True Positive Rate (Recall): 0.8900
False Positive Rate: 0.5800
Precision: 0.6054
F1-Score: 0.7206
AUC: 0.7983

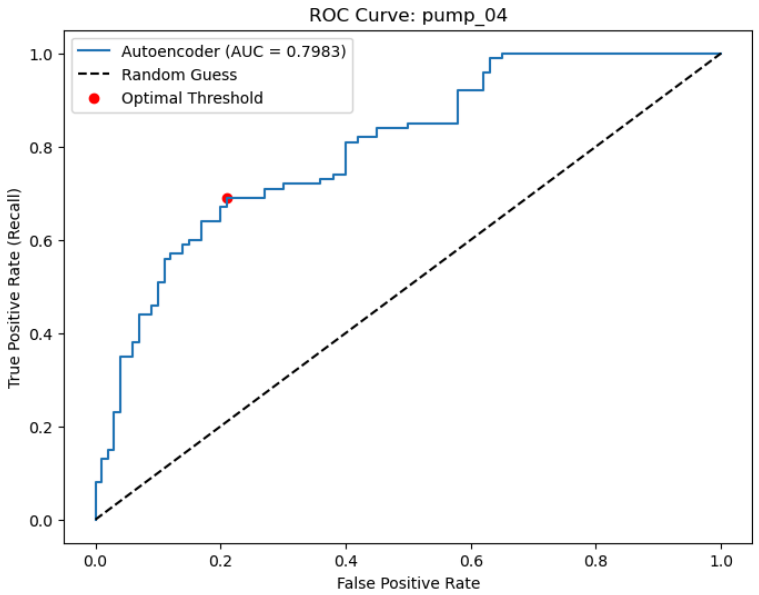
Threshold (at 70th percentile): 15.090943813323975
Accuracy: 0.6950
True Positive Rate (Recall): 0.7200
False Positive Rate: 0.3300
Precision: 0.6857
F1-Score: 0.7024
AUC: 0.7983

Threshold (at 95th percentile): 21.148231697082508
Accuracy: 0.6450
True Positive Rate (Recall): 0.3300

False Positive Rate: 0.0400
Precision: 0.8919
F1-Score: 0.4818
AUC: 0.7983

Using Youden's J statistic

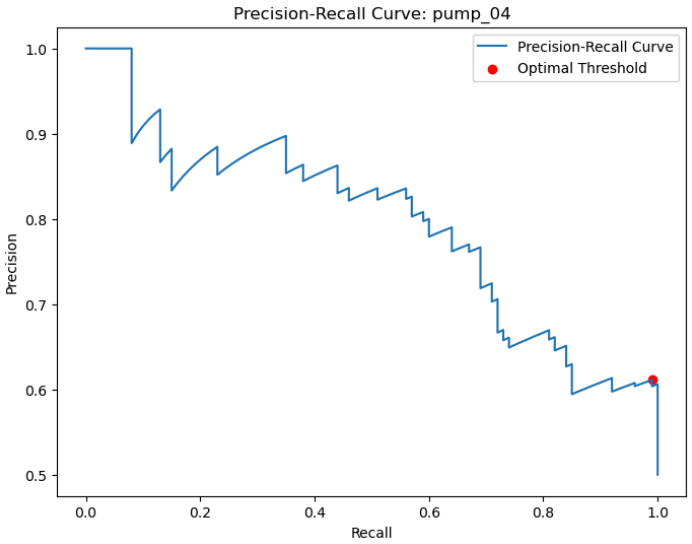
Our second method was to use Youden's J statistic, which is a simple method that balances the true positive rate and the true negative rate. We found it an interesting measure to look at, since it can be useful to find the model that best balances sensitivity and specificity, especially for this specific dataset in which the test data is balanced. The ROC curve with the threshold selected by maximizing the J statistic can be seen below, as well as our evaluation metrics.

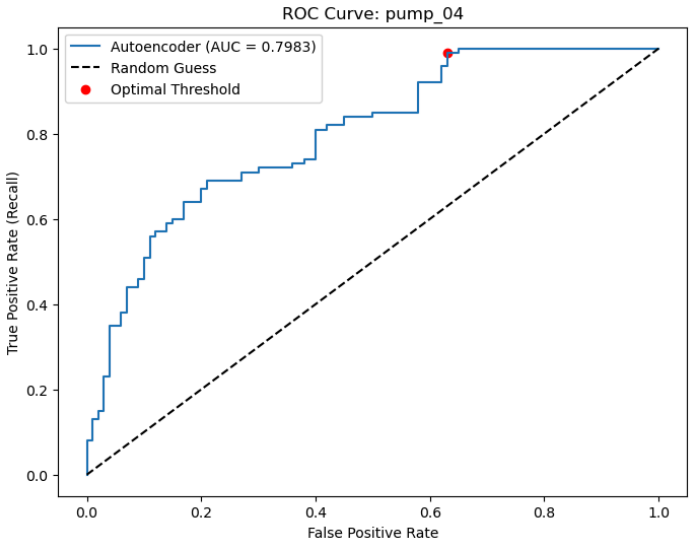


- Accuracy: 0.7400
- True Positive Rate (Recall): 0.6900
- False Positive Rate: 0.2100
- Precision: 0.7667
- F1-Score: 0.7263

Maximizing the F1-score

Finally, and perhaps our most robust threshold (that we would select as being "optimal" for our purposes), is selecting the threshold based on maximizing the F1-score. Our rationale is that it was more important to balance precision and recall than true positive/negative rate, as it is more suited to imbalanced datasets, where having a low number of False Positives is also important (versus a model that is too permissive). Thus while this specific dataset is not imbalanced, we conjecture that the F1-score is generally a more robust metric to maximize for more realistic datasets which will include way more normal data than anomalies. Below we can see both the ROC curve with the threshold with the max F1-score highlighted (right), but also the precision-recall curve, plotting the optimal threshold on it. Both of these plots show that in order to maximize the F1-score we need to move toward a more conservative model, which is also what we can notice in the actual evaluation metrics (we have a recall of 0.99!)



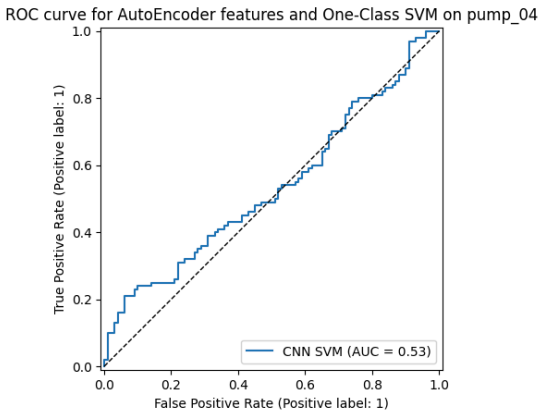


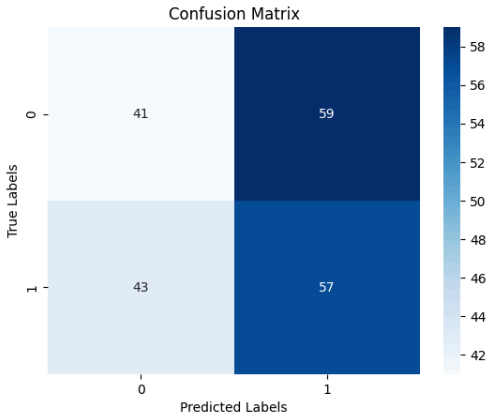
Evaluation Metrics - F1-score maximisation

Accuracy: 0.6800
True Positive Rate (Recall): 0.9900
False Positive Rate: 0.6300
Precision: 0.6111
F1-Score: 0.7557

Bonus question: Change the architecture of the AutoEncoder, to a 2D AutoEncoder using convolutions instead of fully connected layers.

We only had time to try the CNN on SVM but it performed better than the Dense autoencoder.





VALVE

Question 1: AutoEncoder

We trained a symmetric dense autoencoder with 3 linear layers and ReLU activation function after each layer. This basic model didn't perform well, achieving a ROC AUC Score on the test set of 0.3155.

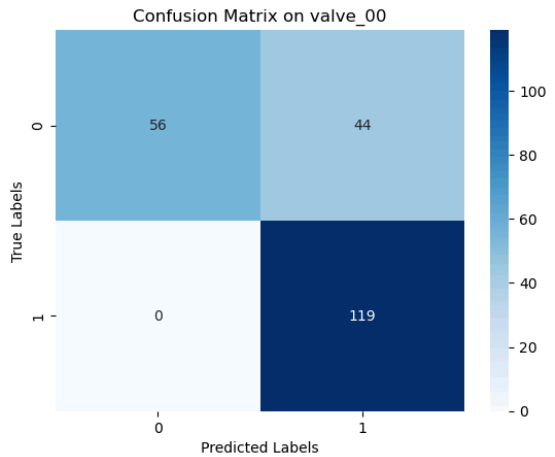
While there could be many reasons for this, one plausible reason is that the AutoEncoder is trained on 100% healthy data, which may lead it to actually learn the features "too well" and generalizing only to normal conditions while failing to recognise any anomalies in the test set. Thus, while the reconstruction error will not tell us much because the extracted features do not represent anomalies well at all, this actually becomes a massive strength when we are using the latent features to train algorithms like isolation forests or one-class SVMs which are designed for this sort of scenario. This might also be why most of our models consistently predict way more anomalies than non-anomalies (can be seen in the confusion matrices, much more activity in the top right quadrant -- false positives -- than the bottom left -- false negatives). This finding actually applies to all of our models!

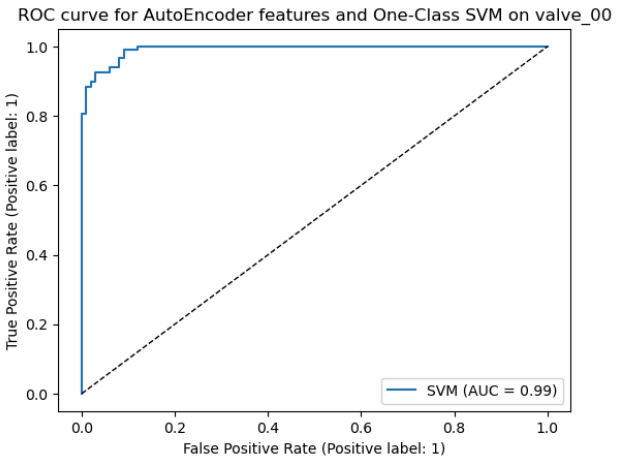
We will discuss thresholding in the 4th section, "AutoEncoder Thresholding." We'll also discuss how we modified the autoencoder architecture.

Question 2: One-Class SVM and Isolation Forest with AutoEncoder features

SVM

Using the autoencoder bottleneck features, we trained a OneClass SVM and achieved an AUC of 0.9918.

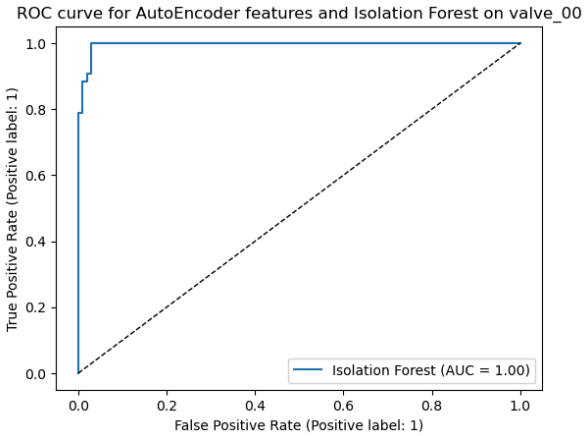


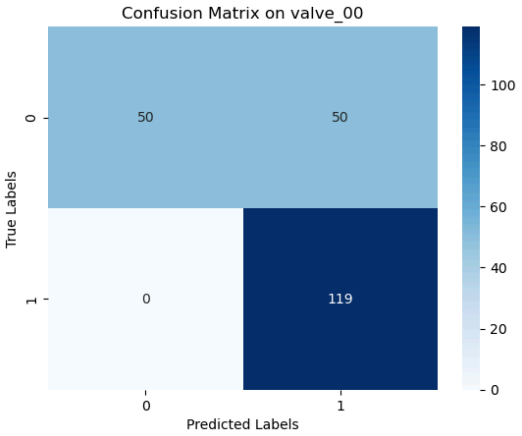


- Accuracy: 0.7991
- True Positive Rate (Recall): 1.0000
- False Positive Rate: 0.4400
- Precision: 0.7301
- F1-Score: 0.8440

Isolation Forest

Using the autoencoder bottleneck features, we trained an isolation forest and achieved an AUC of 0.9958 (rounds to 1!).



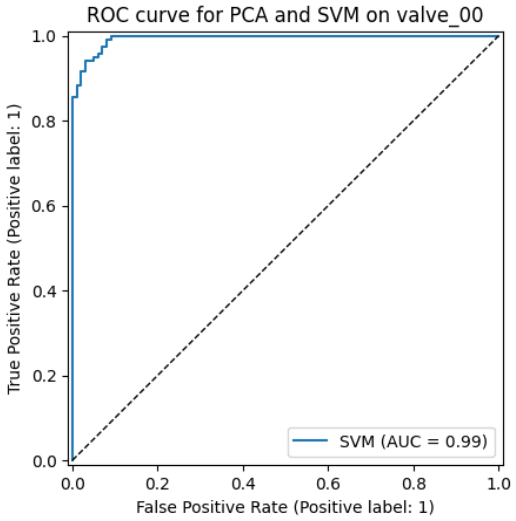


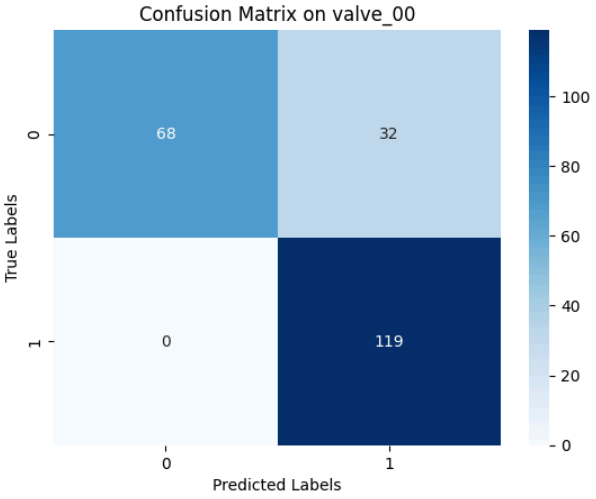
- Accuracy: 0.7717
- True Positive Rate (Recall): 1.0000
- False Positive Rate: 0.5000
- Precision: 0.7041
- F1-Score: 0.8264

Question 3: One-Class SVM and Isolation Forest with PCA features

PCA and OneClass SVM

We fit the PCA model on the training set and use this to transform the train and test set. We used a parameter n_components = 64. We noticed that in the test dataset, it is 54% anomalous data. We achieved a ROC AUC Score of 0.9941.

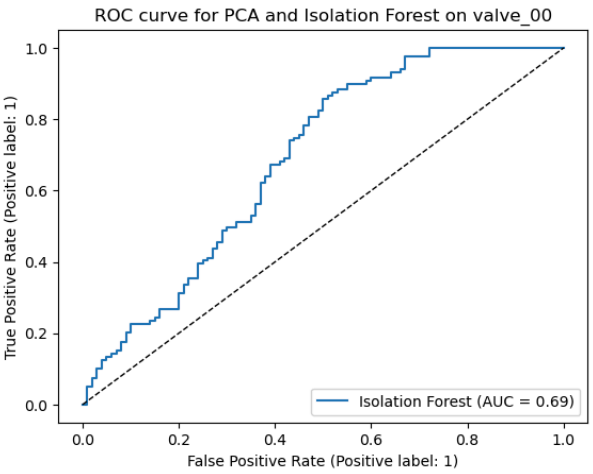


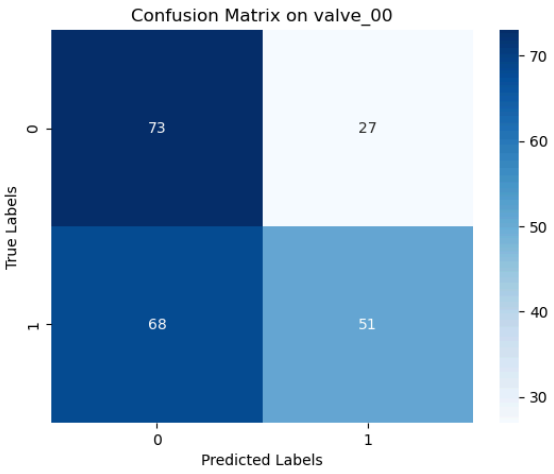


- Accuracy: 0.8539
- True Positive Rate (Recall): 1.0000
- False Positive Rate: 0.3200
- Precision: 0.7881
- F1-Score: 0.8815

PCA and Isolation Forests

We applied PCA to project the MEL-spectrogram into a smaller dimensional space, then trained an isolation forest model on this data. We set the contamination parameter to 0.5 (this represents the proportion of outliers in the dataset), which is the maximum that it can accept. However, from the test set, we know that the contamination is actually closer to 0.54. Our AUC results for the isolation forest are not as good as SVM. This makes sense because Isolation Forest can struggle when there are more anomalies than normal data.



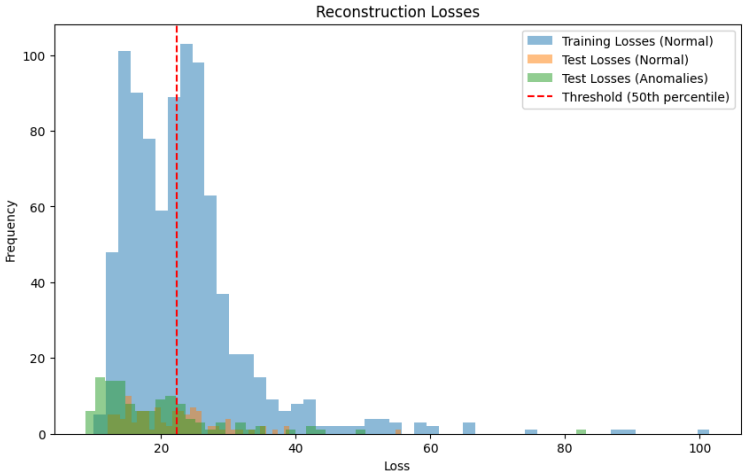


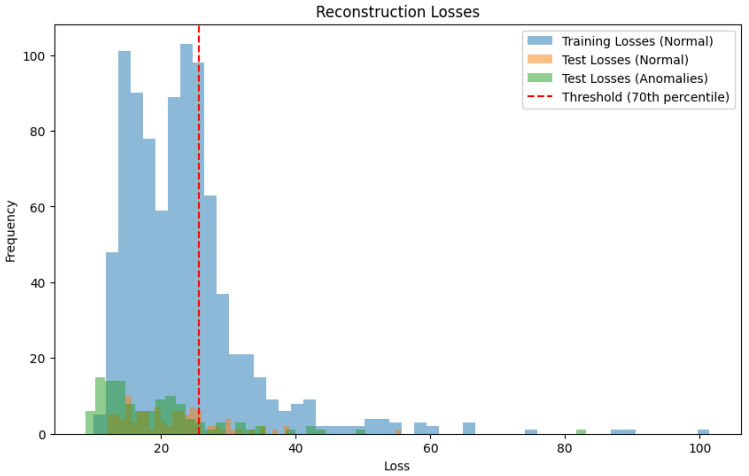
- Accuracy: 0.5160
- True Positive Rate (Recall): 0.3697
- False Positive Rate: 0.3100
- Precision: 0.5867
- F1-Score: 0.4536

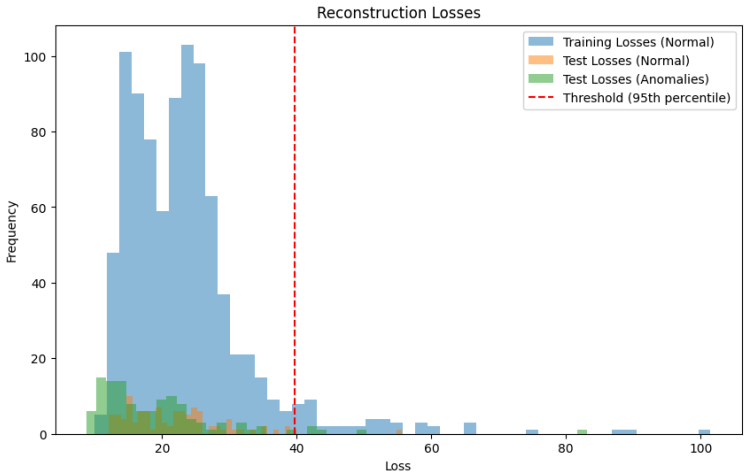
Question 4: AutoEncoder Thresholding

Using the basic architecture of the autoencoder that was provided to us, we don't get good separation of the healthy and anomalous data. We can see that the test losses for normal and anomalies overlap completely, so it's hard to separate with a threshold.

As with the pump, we tried thresholding at the 50th, 70th, and 95th percentile of training losses. However as mentioned in question 1 it is not particularly interesting to find the optimal threshold since the autoencoder performs so poorly (in fact we would be better served simply trying to optimize for some "minimum" F1-score and reversing the prediction!)



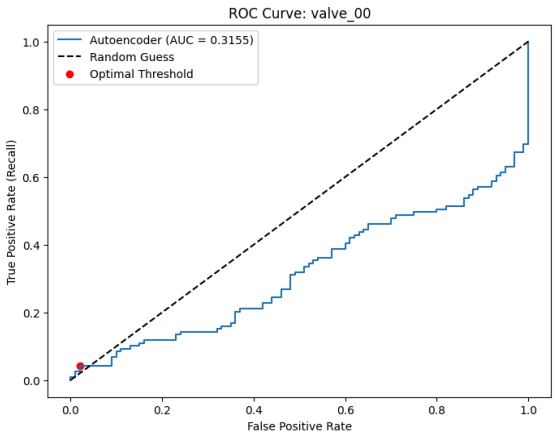


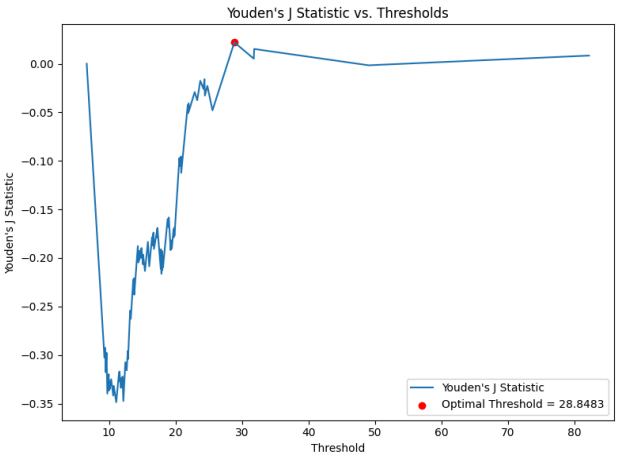


Using Youden's J-Stat method, we found an optimal threshold on the loss of 28.8. Using this threshold, we achieve the following results:

- Accuracy: 0.4703
- True Positive Rate (Recall): 0.0420
- False Positive Rate: 0.0200
- Precision: 0.7143

- F1-Score: 0.0794





Using the F1-score maximization method we have:

- Accuracy: 0.5434
- True Positive Rate (Recall): 1.0000
- False Positive Rate: 1.0000
- Precision: 0.5434
- F1-Score: 0.7041