Ball Bearing Failure Prediction

By: Anand Cheruvu

1. Problem Statement

Rolling element bearings, also commonly known as ball bearings, are the core and vulnerable components in most modern machinery. I plan to develop a prediction model to predict if a ball bearing would fail or not and determine what type of failure it is should it fail.

2. Dataset Description

I used data about ball bearing machines from Case Western Reserve University. There were two types of data that we used. We used the Normal Baseline Data, which served as my ball bearing machine samples that didn't have a fault, and the 48,000 Drive End Bearing Fault Data, which served as my ball bearing machine samples that had a fault, sampled at 48,000 samples per second. There were 6 datasets total, 1 from the Normal Baseline Data, and 5 from the Drive End Bearing Fault Data, with there being 1 dataset for the inner ring and ball faults and 3 datasets for each of the outer ring faults. For the both types of samples we used the data with a motor speed of 1750 revolutions per minute (rpm). These data sets capture the vibration data of the ball bearing machines at both the drive end and fan end. Each row in the dataset represents a time window of vibration data that has been transformed into a set of statistical or time-domain features, such as root mean square (RMS), standard deviation, skewness, and kurtosis etc. These features are useful for reducing raw signal complexity while preserving the information needed to detect abnormalities.

The dataset was modified (shown in table below) as part of feature engineering to include label columns that identify the fault condition of the bearings. These labels may be binary indicators or multi-class identifiers for various fault types such as inner race (IR), outer race (OR), ball fault (Ball), or a normal condition. This labeling approach supports supervised machine learning models, particularly in predictive maintenance applications, to determine not only if a fault exists but also its type and possibly its location.

Furthermore, the structured format of the engineered data—featuring computed features and fault labels—makes it ideal for rapid experimentation with machine learning algorithms. It abstracts away the complexity of working with raw vibration

signals, making it easier to train and evaluate classifiers like Random Forests, SVMs, or deep learning models. The inclusion of a time-related component (either timestamps or sample window IDs) can support temporal analysis or time-series modeling if needed.

	Min	Max	Mean	Std	Var	Skew	Kurtosis	RMS	Crest Factor	Shape Factor	Peak Value	Peak to Peak Value	Impulse Factor	Zero Crossing Rate	Fault
0	-2.29310	2.36904	0.00854	0.76247	0.58137	0.05417	0.15214	0.00854	277.49783	0.01499	2.36904	4.66214	4.16019	0.10875	Inner_Ring
1	-2.43955	2.64462	0.01081	0.70667	0.49939	0.09202	0.61442	0.01081	244.54560	0.01993	2.64462	5.08417	4.87362	0.10753	Inner_Ring
2	-2.58746	2.68196	0.00603	0.69654	0.48517	0.05456	1.14844	0.00603	445.12876	0.01198	2.68196	5.26942	5.33083	0.11118	Inner_Ring
3	-3.13716	3.41775	0.01450	0.85414	0.72955	0.08097	0.89554	0.01450	235.69159	0.02215	3.41775	6.55490	5.22151	0.10753	Inner_Ring
4	-2.87034	2.76979	0.00384	0.78338	0.61368	0.05333	1.19655	0.00384	747.85569	0.00664	2.87034	5.64013	4.96659	0.10693	Inner_Ring
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2327	-0.18655	0.25374	0.03062	0.06689	0.00447	-0.00099	0.03221	0.03062	8.28707	0.52410	0.25374	0.44029	4.34329	0.18104	Normal
2328	-0.17669	0.23689	0.03178	0.06661	0.00444	0.03104	-0.23684	0.03178	7.45463	0.53852	0.23689	0.41358	4.01444	0.19259	Normal
2329	-0.16621	0.26504	0.03160	0.06708	0.00450	0.26562	-0.02273	0.03160	8.38612	0.54503	0.26504	0.43125	4.57072	0.18469	Normal
2330	-0.18327	0.23771	0.03038	0.06447	0.00416	0.16218	-0.32397	0.03038	7.82357	0.53187	0.23771	0.42098	4.16110	0.18469	Normal
2331	-0.17320	0.30674	0.03063	0.06666	0.00444	0.27937	0.21863	0.03063	10.01434	0.53345	0.30674	0.47994	5.34218	0.19684	Normal

Table 1. Feature Engineered Dataset

3. Approach

The following diagram illustrates the five step approach I took for this project.

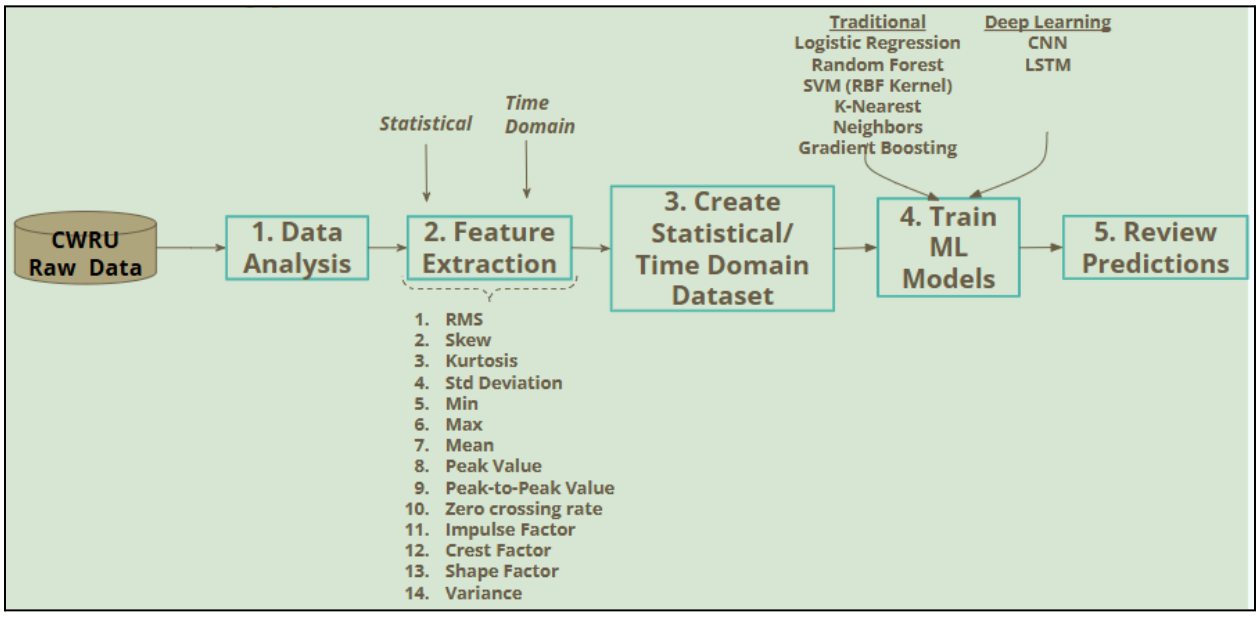


Figure 1. Approach

4. Experiment Results

Here is a summary of the results from my project:

Summary of ML and DL Model Performance Metrics

Model	Avg Confusion Matrix Entries	Accuracy	True Positive Rate	False Positive Rate	Area under ROC Curve (AUC)
Logistic Regression	36.4375	0.94168	0.94173	0.01944	0.99511
Random Forest	36.4375	0.9777	0.97773	0.00743	0.99859
Support Vector Machine	36.4375	0.97084	0.97089	0.00972	0.99832
K Nearest Neighbors	36.4375	0.97427	0.97432	0.00858	0.99477
Gradient Boosting	36.4375	0.97256	0.97256	0.00915	0.99756
CNN	36.4375	0.95883	0.95889	0.01373	0.9972
LSTM	36.4375	0.96226	0.96232	0.01258	0.99694

The confusion matrices and ROC curves for the models are shown below sections.

5. The Algorithms

I used 7 algorithms for this project, 5 of them being traditional machine learning models and 2 of them being deep learning models. The 5 traditional machine learning models are:

- Logistic Regression
- Random Forest
- Support Vector Machine
- K Nearest Neighbors
- Gradient Boosting

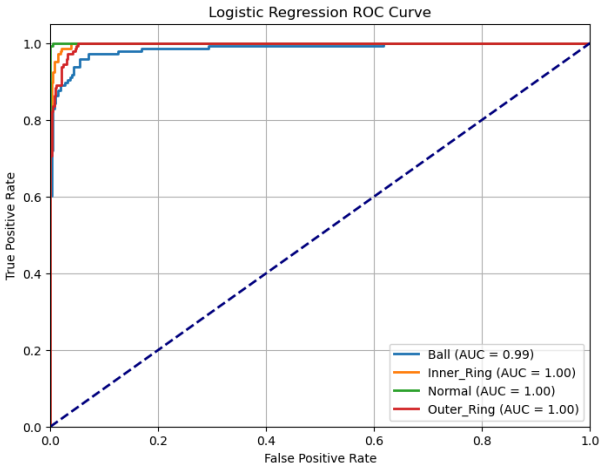
My 2 deep learning models are Convolutional Neural Networks(CNN) and Long Short Term Memory(LSTM).

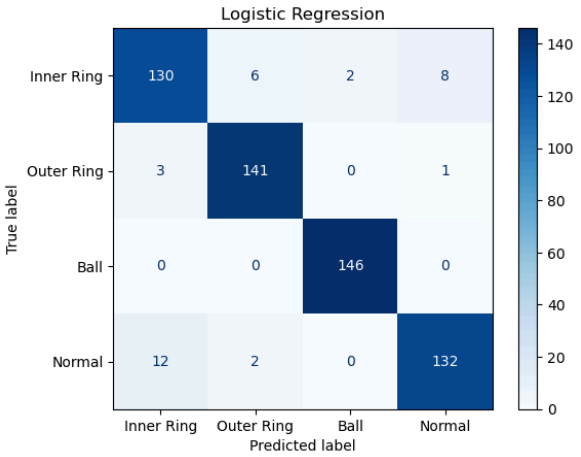
I analyzed the ROC curve (as shown in the sections below) for all models. All models demonstrate exceptional performance with AUC (Area Under Curve) values ranging from 0.99 to 1.00 across all classes. The LSTM, CNN, Gradient Boosting, Random Forest, and SVM models achieve perfect or near-perfect classification for all four categories. K-Nearest Neighbors performs slightly lower on the Outer_Ring class (AUC = 0.99), while Logistic Regression shows comparatively more gradual curves for the Ball class but still maintains high overall performance. Each graph includes a diagonal dashed line representing random chance classification (AUC = 0.5), highlighting how significantly better these models perform than random guessing. The high AUC values

across all models suggest this is likely a well-separated classification/prediction problem where the features provide strong discriminatory power.

5.1 Logistic Regression

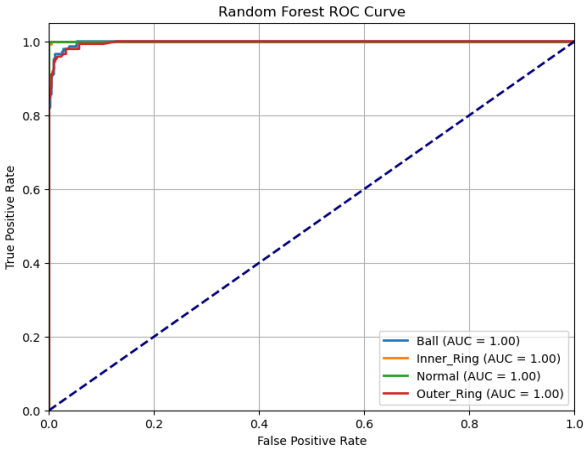
I picked Logistic Regression as it is a simple and powerful baseline classifier. In terms of my ball bearing machine problem it allowed me to create easily interpretable models since features like the root-mean-square(RMS), standard deviation, and skewness are cleanly separated between faulty and non-faulty conditions.

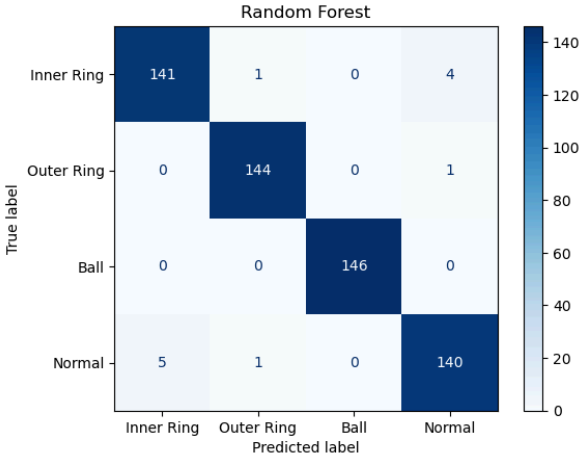




5.2 Random Forest

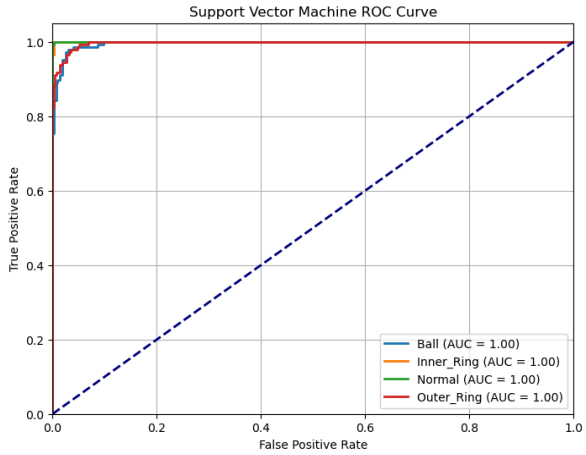
With bearing faults different features may combine in complex ways that Random Forests are good at handling. Random Forest is also robust to noise and handles overfitting well which is helpful when considering all of my features.

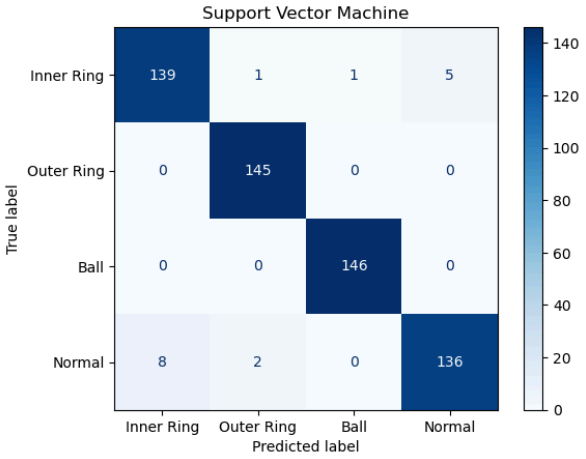




5.3 Support Vector Machine (SVM)

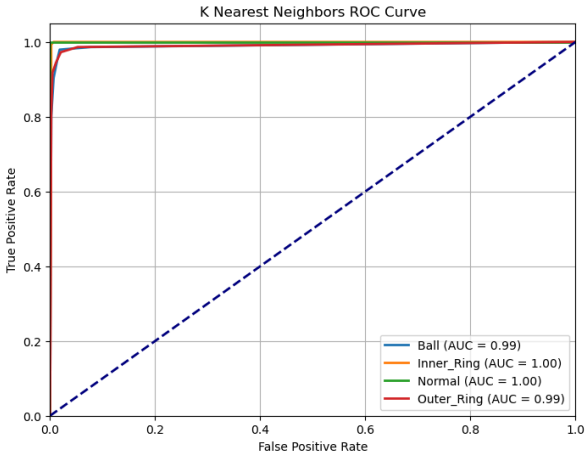
Support Vector Machines can model nonlinear separations using different kernels (in my case the RBF kernel), serving as a nonlinear classifier that can be used for my vibration data that might not be linearly separable.

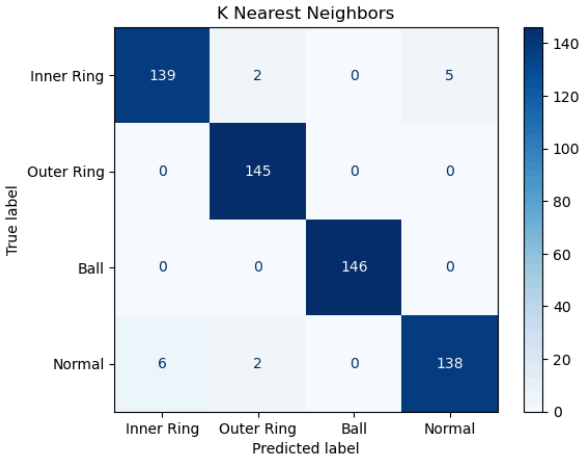




5.4 K-Nearest Neighbors (KNNs)

My Bearing Fault Data tends to produce locally clustered patterns of my features. KNNs are effective for clustered feature patterns while being an intuitive baseline classifier, serving as a quick and relatively lazy learning approach model.

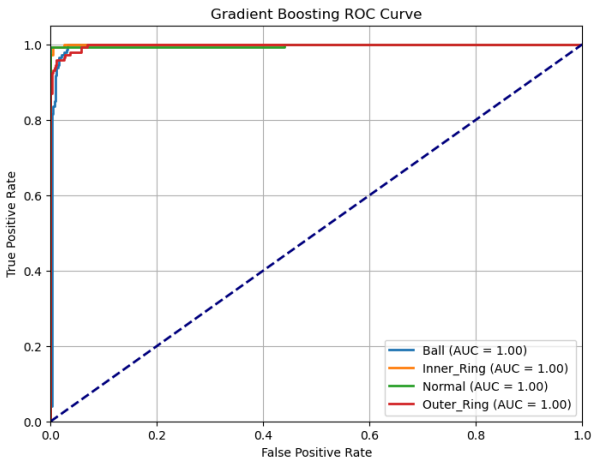


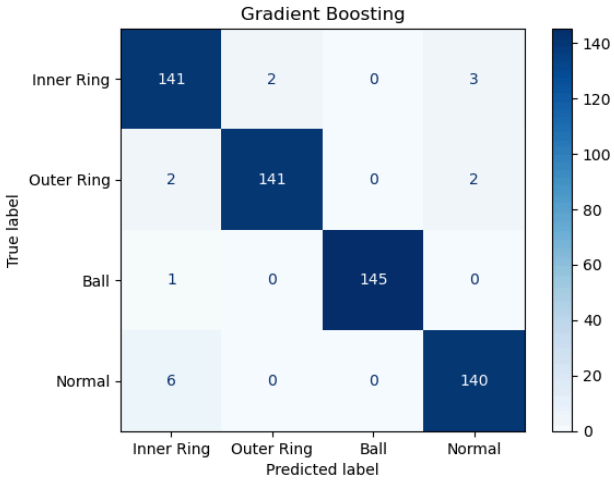


5.5 Gradient Boosting

Gradient Boosting combines weak learners over multiple iterations to create a strong classifier. When properly tuning its high amount of customization it is very effective at

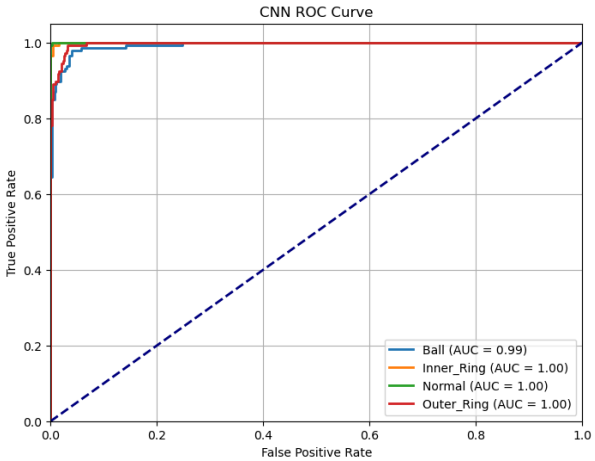
capturing small variations in feature behavior. While it can have longer training times its high predictive power made it a valuable choice for my project.

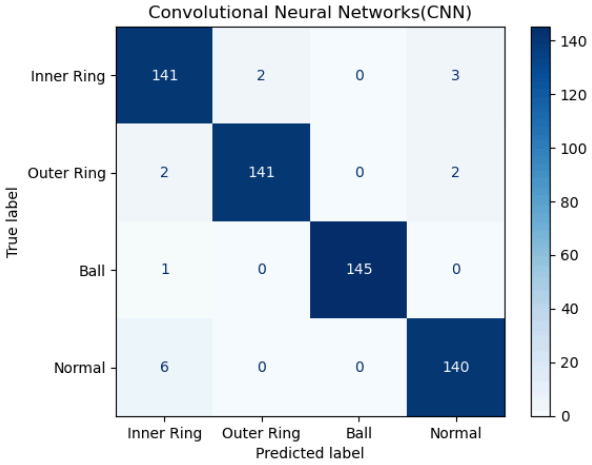




5.6 Convolutional Neural Networks (CNN)

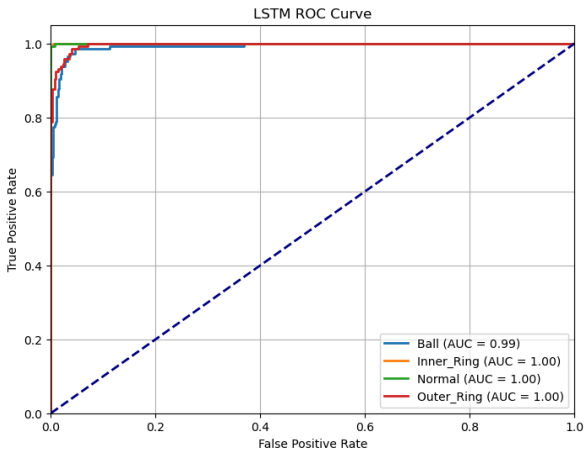
Though CNNs are typically known for their ability to process image data they are also excellent at learning local patterns. In my case this includes sudden spikes in vibration data. By reshaping the vibration features into “feature maps” CNNs can automatically learn patterns without manual feature extraction.

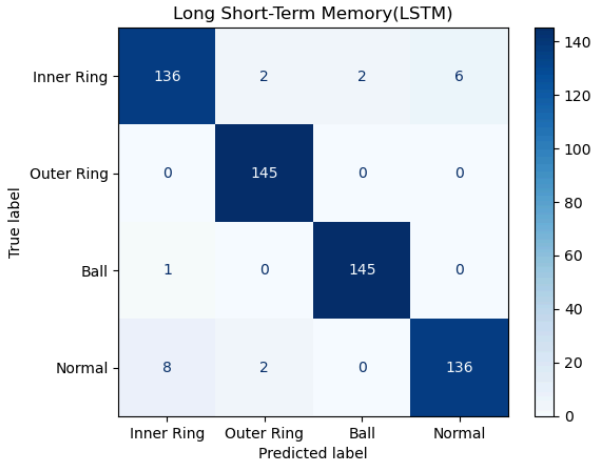




5.7 Long Short-Term Memory (LSTM)

LSTMs are excellent at capturing sequential data over time. This is helpful as for bearing degradations, early faults can gradually evolve. Capturing the temporal evolution is important for understanding the faults and being able to predict early faults before they become a bigger issue.





6. Results and Conclusion

Overall, the models were **very successful in predicting both faulty and non faulty ball bearings**. All of my traditional machine learning models have an accuracy of 97 percent, with Logistic Regression being the exception at a still high 94 percent whereas the deep learning models both have an accuracy of 96 percent. Despite the Deep Learning Models having an AUC score higher than most of the traditional models, **Overall traditional models were more successful.** This is likely due to the limited data size, absence of raw signal sequences, and shallow network layers. This is because deep models thrive on large amounts of training data, temporal/spatial patterns, and more layers/tuning to optimize their performance.

6.1 Traditional Model Conclusions

Logistic Regression, SVM, Random Forest, KNN, and Gradient Boosting are all commonly used for classification tasks. Their performance depends heavily on the dataset characteristics. In this specific case, the accuracy scores and confusion matrices provide insights into their relative performance.

Accuracy: Higher accuracy suggests a better predictive ability. Looking at the results table we can see that all of the traditional machine learning models have a high accuracy, meaning they all likely have great predictive ability. Looking at that table we can also see that Logistic Regression has an accuracy lower than even the deep learning models. This is likely due to how Logistic Regression handles the datasets compared to the other models. Logistic regression works best on simple, linearly separable datasets. Its lower accuracy relative to the other models suggests that the

data is not simple or it is not linearly separable, perhaps even having complex interactions between features. The other models having a higher accuracy also imply these facts about my dataset as they are better at handling complex data that isn't necessarily linearly separable.

True positive rate and false positive rate: The true positive rate mirrors the accuracy both in their relative values and what they tell us about the model's performance against my dataset. The false positive rate similarly mirrors accuracy and the relationship between the models but it is inversed, with Logistic Regression having the largest false positive rate.

AUC scores: My AUC scores for the traditional models have a slightly different distribution than my previously discussed metrics. KNNs have the lowest AUC score, being the only traditional model along with Logistic Regression and Gradient Boosting(barely) to have a lower AUC score. This is likely because KNN doesn't produce true probability scores but rather counts the amount of occurrences in a class among neighbors, which limits its ability to rank its predictions, which is what AUC measures. Logistic Regression has a lower score likely due to its inability to assign well-separated probability scores for datasets that aren't linearly separable, once again pointing to the fact that this dataset isn't completely linearly separable. The other models have higher AUCs due to their abilities to separate classes well and have reliable confidence estimates of the output probabilities. Looking at the ROC curves also back what the AUC scores tell us as KNN and Logistic Regression are less steep before flattening out compared to the other models, mirroring their lower AUC score. In general the AUC scores are all very high reflecting that all of the models rank predictions well.

6.2 Deep Learning Model Conclusions

Deep learning models often require very large amounts of data and are typically used for complex pattern recognition including image recognition or natural language processing. Their lower accuracy and above average AUC score tell us that in general while these models are better at separating classes they aren't as good at the specific threshold used for accuracy.

Accuracy: CNN and LSTM's lower accuracy compared to the traditional models suggest that there wasn't as much data as these models would have liked to train on. This is compounded with the fact that I did extensive feature engineering before training these

models, leading to them attempting to directly learn features that have already been manually extracted.

True positive rate and false positive rate: As with my traditional learning models these mirror accuracy in how they are distributed, inversely in the case of the false positive rate. This is because my models make few gross misclassifications, leading to a high true positive rate, low false positive rate, and high accuracy, which makes these values appear to mirror each other.

AUC scores: The Deep Learning Models above average AUC relative to the traditional models contrasts with its accuracy in relation to the same models. This difference comes from the Deep Learning Models learning complex patterns better, leading to them often assigning higher scores to true positives and lower scores to negatives which improves ranking quality leading to a better AUC. Another reason is that since AUC disregards the threshold that accuracy focuses on it leads to a higher AUC. Looking at the ROC curves for my Deep Learning Models we see that they are steeper than the models that have a lower AUC, mirroring these models above average AUC.