My name is Neil Kloper and in this video, I will explain the process I used for my submission to the ODSC Data Science Hackathon, as well as my findings in the dataset used, and some key takeaways from the experience.

The Dataset used for this Hackathon is the ‘Electric Motor Temperature’ dataset from Paderborn University. The Dataset is already scaled by Z-score, so scale and range between elements are extremely similar. For visualizations, I chose to stick primarily with bivariate plots using seaborn’s pairplot class, and occasional heatmaps.

To begin the notebook titled ODSC\_SKlearn.ipynb has my exploration of the data, attempts at feature engineering, and initial baseline. I began by overviewing a basic pairplot, and then diving deeper by creating a new dataset with the target variables changed into a binary value representing which side of the value’s mean it fell. While there were some clusters that easily were distinguishable, the majority of the datapoints were not linearly separable for any value. I chose to use models that made inference based on individual rows instead of seeing each row as an element in a time series (which did hinder my performance). When splitting the data for training and validation I discovered than doing a random split created models that overfit the training and validation data (which is common in time series), splitting by index location (setting the most recent 20% aside) created models which had a closer validation and test score, later when I was running tests with neural networks I was able to use random splits and still maintain a reasonable delta between the validation score and test score. As one last test in the notebook, I created a function to calculate the permeation importance (a feature ranking method) and discovered that the majority on the signal was only coming from 3 features, the ambient temperature, the coolant temperature, and d component current.

Permeation importance works by testing a model on the same dataset multiple times, each time it will select one feature and randomize the values and rerun to determine which features add signal and which contribute mostly noise. There is already a module for feature importance Eli5, however, its been having some issues which depreciations of some of its dependencies so I’ve been meaning to write my own function for a while.

The FastAI notebook sets up neural nets for tabular data. This notebook isn’t necessarily meant to be run in order. As you will notice by the commented out code blocks, I used a variety of methods to train the models contained in these notebooks. In some cases validation set of the data was selected at random, other times the validation section was selected by splitting the data based on a point in time (typically the final 20% of the data). I trained many different models, with several different architectures so that I could see what capacity the model needed to match the complexity of the problem. Most of the code I used was fairly boilerplate, nothing overly fancy. Of the target features,’pm’, or permanent magnet temperature, had the greatest complexity to it, initial 2 layer models were very ineffective at estimating this feature, once I started using models with 5 or more layers, the performance greatly improved. The goal is to create a blend of models, which when they have their predictions averaged, produce a decent score.

The blend resulted had an RMSE score of around .47, where the best individual model in the blend could achieve a score of .45. \*However\*, the mean absolute error of the blend (the average distance the prediction has from the actual value) went from a score of .35 for individual models down to .21. To put that into perspective. if a model only guessed the mean value for each entry, it would score .91 for RMSE and .83 for MAE.

There is room for improvement, I know there are better architectures for this challenge, I think semi-supervised learning might have generated a better performing model, I also was unable to get alternate loss functions to work, and I think I could have improved things a smidge by doing some data augmentation (yes you can do this with tabular data).

I had fun with this competition, and it has helped me get a clearer idea of what I want to focus on next in my machine learning journey.