W207 Final Project: Stock Prediction Models

John Andrus, Luis Chion,
Kevin Fu, Fernando Roriz

Motivation

- Stock price prediction has been a major challenge in finance.
- ‘Beating the market’ is a long term goal in the field.
- Machine learning can be a powerful tool to explore new approaches to the market data.
- Which ML models can generate the highest returns in the stock market?

Literature

- The literature has established that stock market prices do not follow random walks and the variability of stock returns can be explained by factors such as market risk, size and growth (Lo and MacKinlay (RFS, 1988), Fama and French (JF, 1992) and Fama and French (JFE, 1993)).

- Cochrane (RFS, 2008) shows that even small out-of-sample R2 (from 4% to 7%) has large *economic* significance and variables such as *dividend-growth ratio* are relevant predictors.

- Gu, Kelly and Xiu (RFS, 2020) runs a comparative analysis of machine learning methods and shows that best performing methods to measure asset risk premia are trees and neural networks. These models trace their predictive gains to allowance of nonlinear predictor interactions that are missed by other methods.

- At this time, many papers are being produced and published trying to apply machine learning techniques to predict stock markets and this project try to add on this strand of the literature.

Jane Street Market Prediction - Kaggle

- In this competition, for a set of possible trades presented, we have to decide if should add this trade to our portfolio or not.

- Roughly, our main goal is to add to our portfolio trades that have positive returns and pass the ones with negatives returns.

- For each possible trade, we are presented with 130 features and can use them to decide if this will have a positive return or not.

Jane Street Market Prediction - Kaggle

- The return of a trade is given by the ‘resp’ variable and ‘weight’ represents the leverage for that specific trade. Finally, the ‘action’ variable is our binary target that takes value 1 if *resp* > *0*, and 0 otherwise.

$$p_i = \sum_j (weight_{ij} * resp_{ij} * action_{ij})$$

- The competition is evaluated on the following utility score:

$$u = min(max(t, 0), 6) \sum p_i \quad \text{,where} \quad t = \frac{\sum p_i}{\sqrt{\sum p_i^2}} * \sqrt{\frac{250}{|i|}}$$

- $|i|$ is the number of unique dates in the test set.

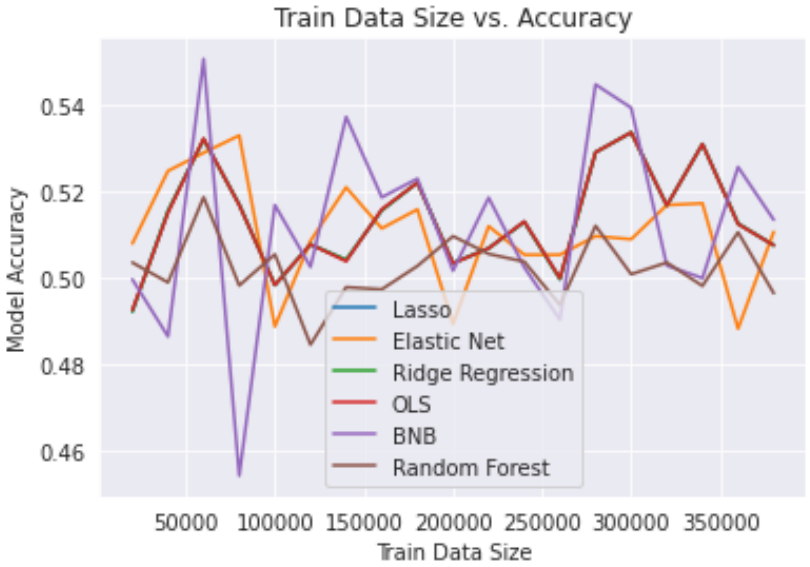
Data: Source, Description, Stats

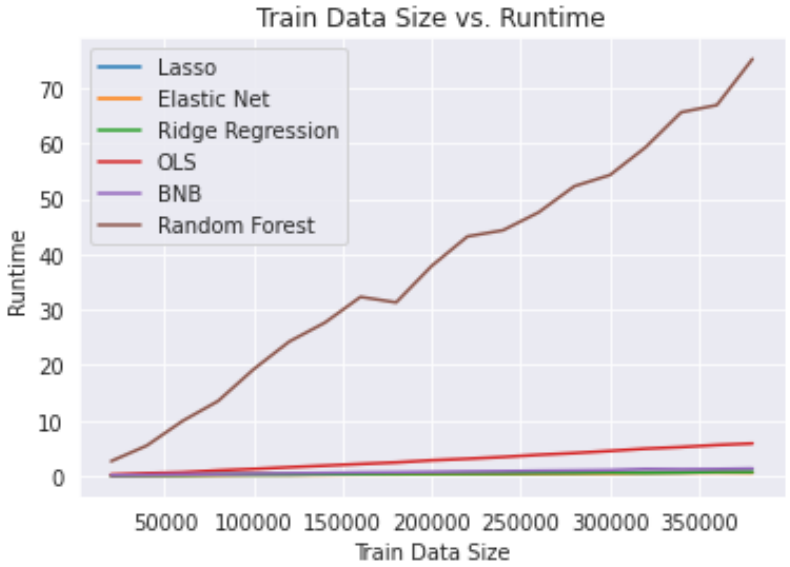
- Data
  - ~1.6 million observations of 130 anonymized features
  - Removed highly corr. features
  - Raw data 18% null, nulls replaced with median (zeros)
- Binary Labels (Target Variable)
  - 1 - Make a Trade
  - 0 - Take No Action
- Named Features
  - Date & ts_id : Integer date and timestamp
  - Weight and resp : Together represent return on a trade

Model Evaluation

- **Horse Race** across a variety of machine learning techniques:
  - Random Forest
  - K-Nearest Neighbors
  - Naive Bayes
  - OLS
  - OLS with PCA
  - Logistic Regression with PCA
  - Ridge Regression
  - LASSO
  - Elastic Net
  - LSTM
  - GMM
  - CNN

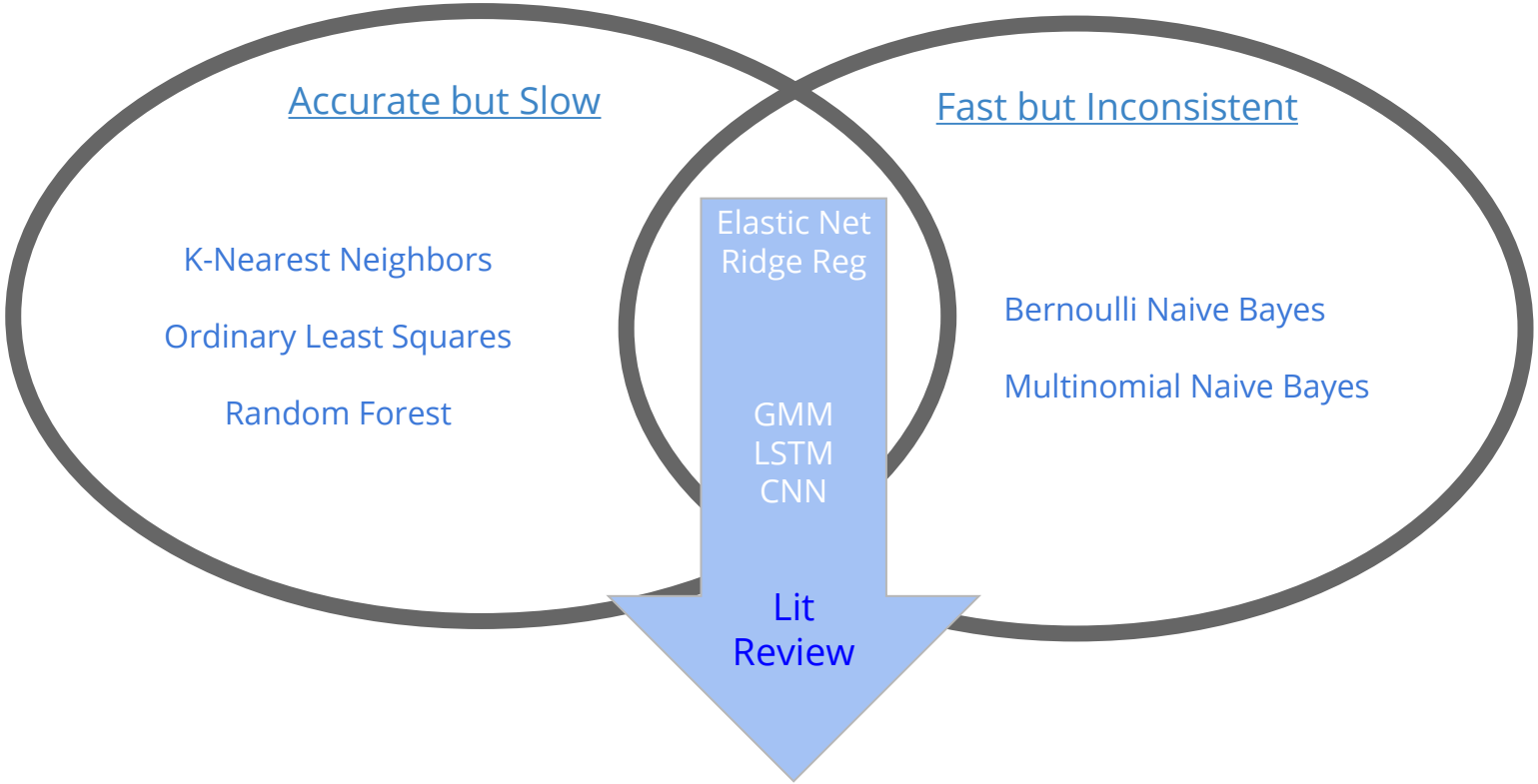
Model Selection: Accuracy vs Runtime





*K-Nearest Neighbors excluded due to extremely long runtimes

Model Selection

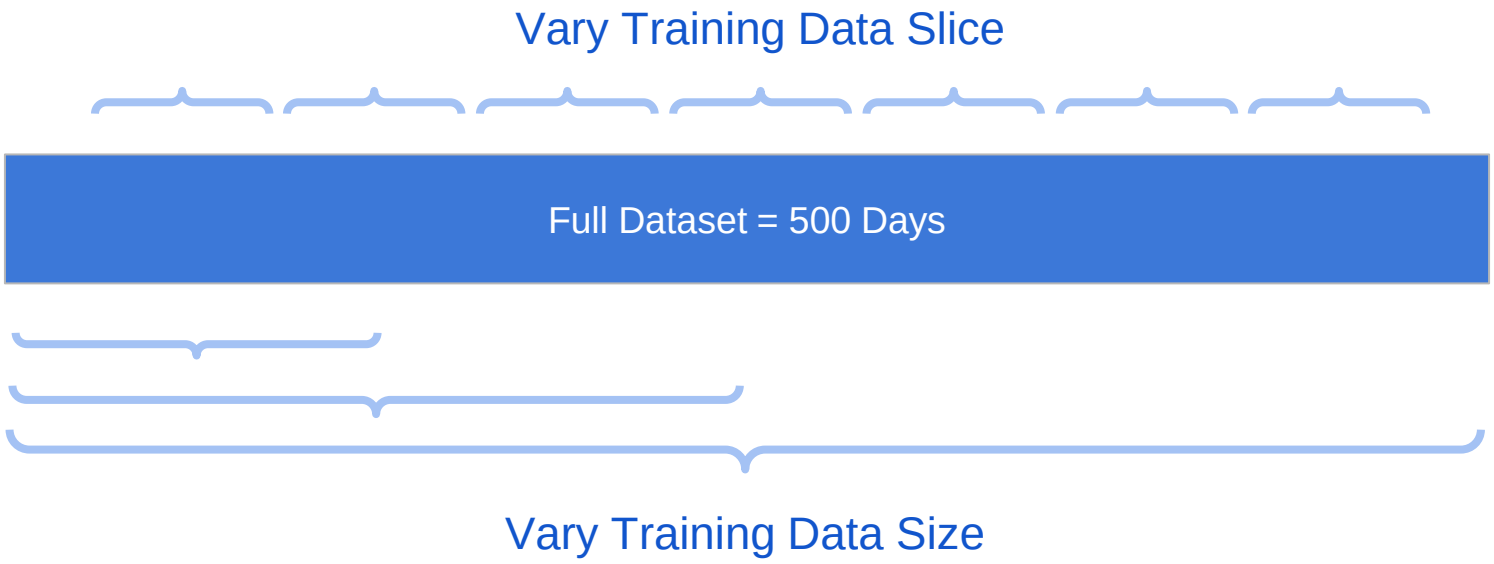


Gaussian Mixture Model (GMM) - Hyperparam Tuning

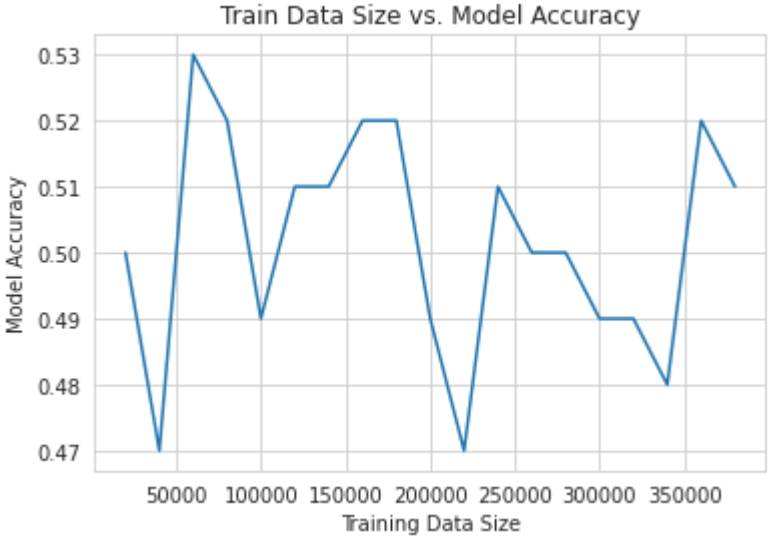
- Performance maximized with 3 GMM Components
- Performance best with tied covariance
- Improvement potential by implementing PCA

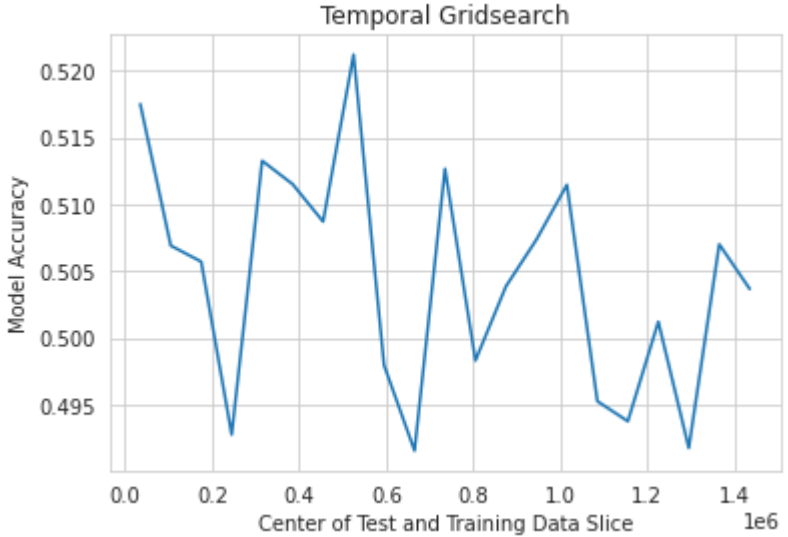
Initial Model
GMM Components = 3
Training Data Size = 50,000
Model Accuracy = 52.83%

Evaluating Consistency of Model Performance



Accuracy Variation by Size and Slice

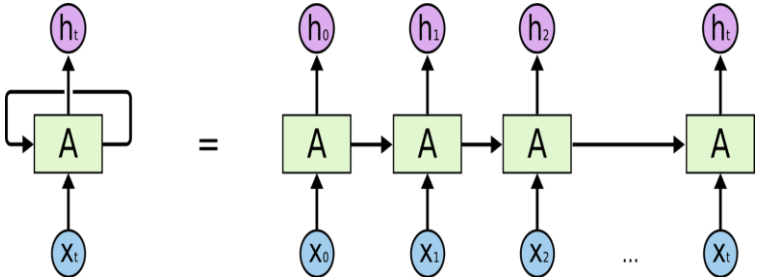




Summary

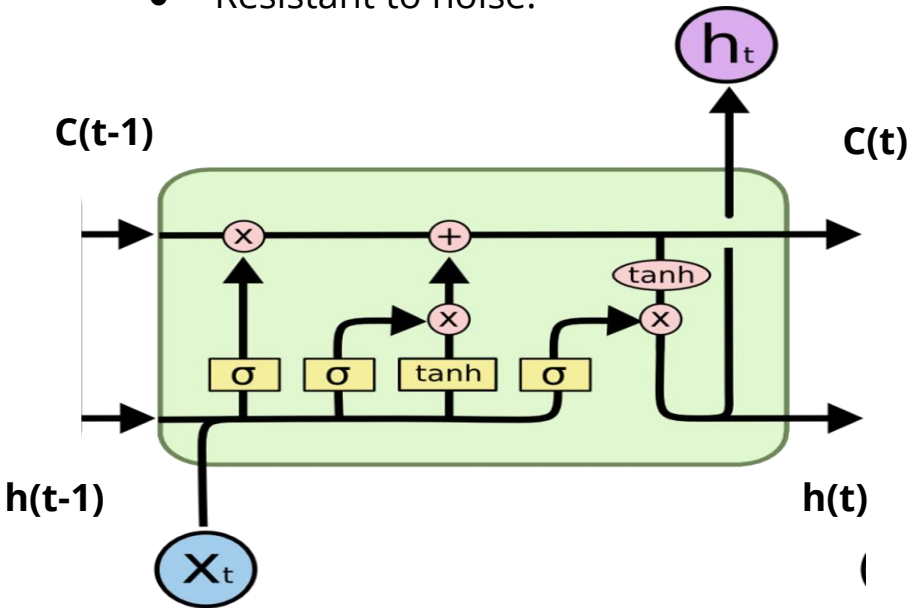
- Preliminary success with 3 Component GMM Models
- PCA resulted in reduction of model accuracy below 50%
- Inconsistent Performance
- Very low utility score

Recurrent Neural Networks (RNN) - Long Short Term Memory (LSTM)



- Stores information from the past for an arbitrary length of time.
- Resistant to noise.

- Memory cells replace hidden layers
- 4 neural network layers: 3 sigmoid, 1 tanh
- Cell state “C(t) - conveyor that adds and subtract information
- 3 gates: forget, input and output



LSTM in depth

Financial Markets

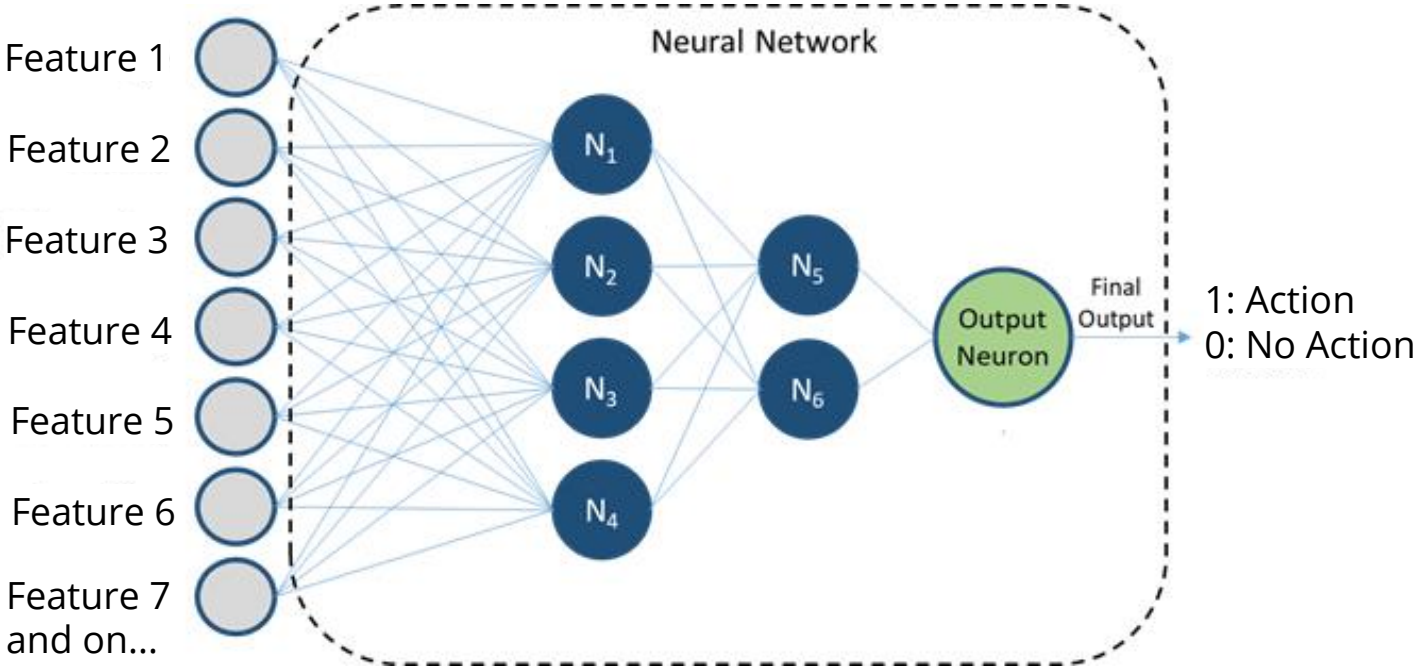
- Market prediction as key activity for major financial institutions
- Econometric (ARIMA,GARCH) or risk-neutral models are less effective with regime changes
- Random nonstationary and noisy sequences
- Avoid long-term dependence.
- Good non-linearity and self-learning

Model Setting

- 4 layers each with a dropout layer
- Final layer with a sigmoid activation function
- epochs = [500, 2000, 5000]
- batch_size = 3500 (1 week - 500 trades/day)

	model	accuracy	utility
0	LSTM 500 epochs 500 batch size	0.50374	290.263
1	LSTM 2000 epochs 500 batch size	0.50258	351.439
2	LSTM 100 epochs 3500 batch size	0.504	314.974
3	LSTM 500 epochs 3500 batch size	0.5046	277.411
4	LSTM 2000 epochs 3500 batch size	0.50448	323.897
5	LSTM 5000 epochs 3500 batch size	0.5029	302.088
6	LSTM 100 epochs 7000 batch size	0.50214	318.852
7	LSTM 500 epochs 7000 batch size	0.50494	376.863

Convolutional Neural Network (CNN) - Overview



Hyperparameter Tuning

Improves Model

- Kernel sizes
- Batch sizes
- # of epochs
- Type of activation function

Does Not Improve Model

- PCA
- # of nodes in density layer
- # of nodes in convolutional layers
- Training sizes

Results

- Performance maximized through trial and error
- Used variety of different kernel, density, batch, epoch sizes
- No one model consistently outperformed

"Best" Model over 12 Iterations

Model Accuracy = **51.1%**

Utility Score = **474.95**

Kernel Size = 1

Density Size = 2

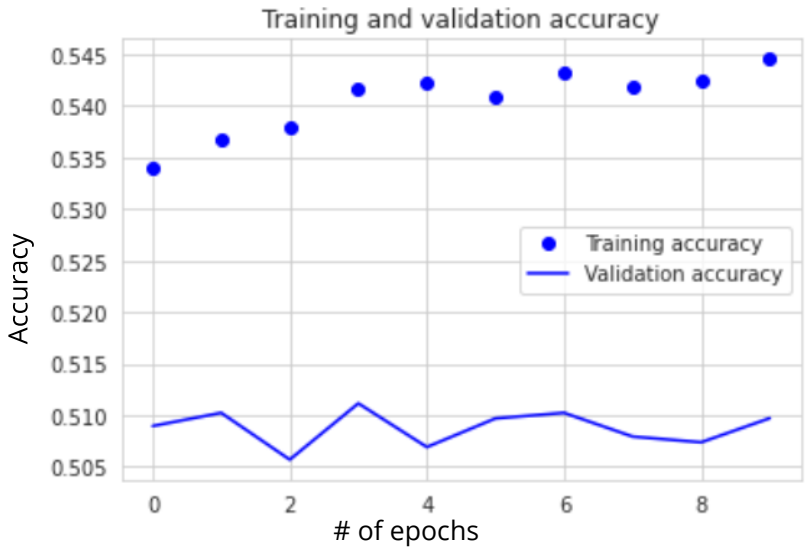
Batch Size = 50

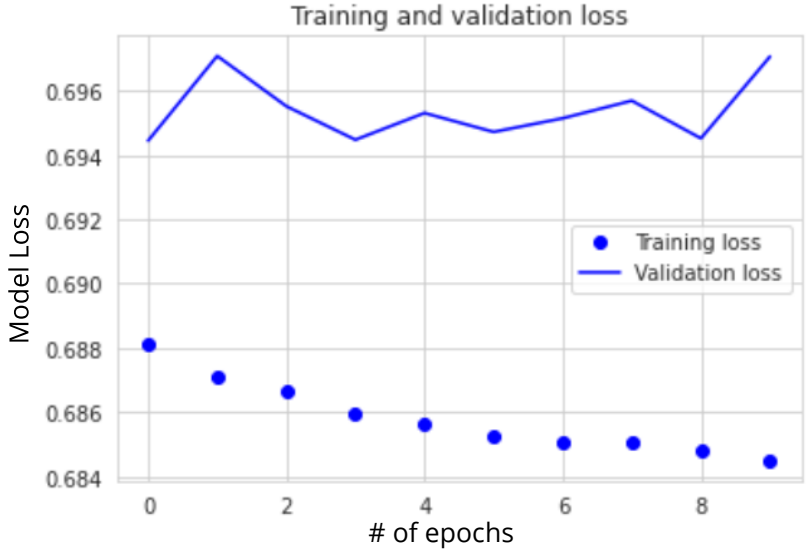
Epoch Size = 10

	model	accuracy	utility	kernel_size	density_size	batch_size	epoch_size
0	CNN Model	0.50768	107.667	1	2	50	3
1	CNN Model	0.51114	474.947	1	2	50	10
2	CNN Model	0.51014	456.733	1	2	3500	3
3	CNN Model	0.5105	415.865	1	2	3500	10
4	CNN Model	0.50914	104.078	3	2	50	3
5	CNN Model	0.50598	150.877	3	2	50	10
6	CNN Model	0.49746	151.303	3	2	3500	3
7	CNN Model	0.49864	116.028	3	2	3500	10
8	CNN Model	0.50872	456.647	5	2	50	3
9	CNN Model	0.50666	383.929	5	2	50	10
10	CNN Model	0.50252	406.793	5	2	3500	3
11	CNN Model	0.50174	384.838	5	2	3500	10

Results: “Best” Model

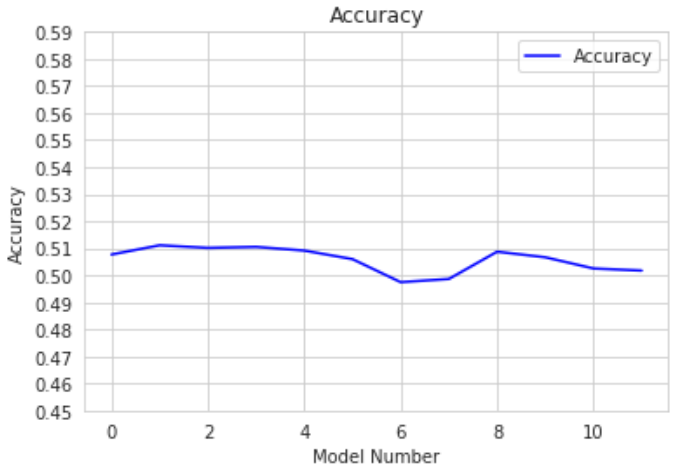
- Accuracy and validation loss are consistent (little variance) over epochs,
- Generally does not improve with additional iterations (ceiling @ 3 epochs)

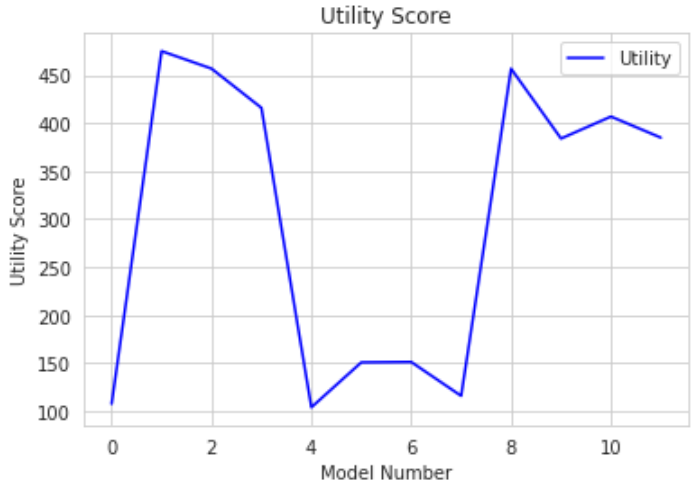




Takeaways

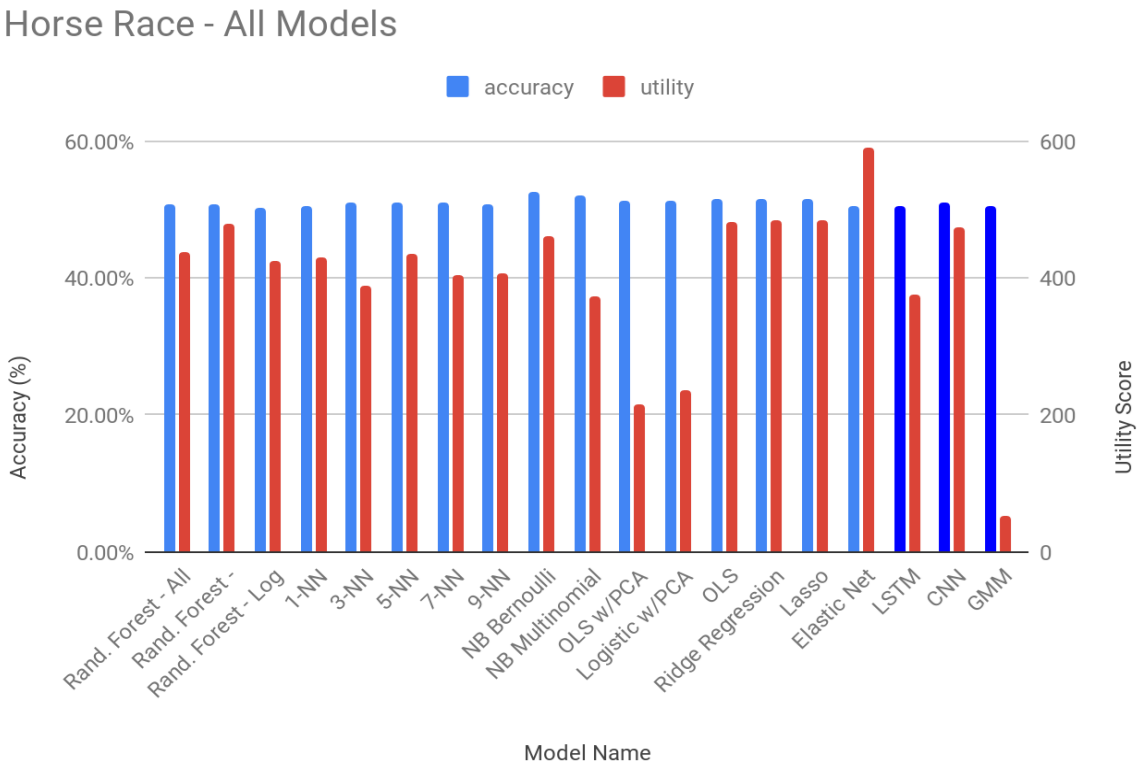
- Accuracy and Utility Score in-line with other models
- Low variance in accuracy
- High variance in utility score





Conclusion

- In-line performance with other models
- More consistent in terms of accuracy but not utility



Appendix

Autoencoder Multi Layer Perceptron (MLP)

- Steller, too good to be true numbers
- Consistently generates 80%+ accuracy, 3000+ utility
- High possibility of overfitting

	model	accuracy	utility
0	Supervised Autoencoder Multi Layer Perceptron (MLP)	0.90848	5369.87

Autoencoder Multi Layer Perceptron (MLP)

- Training and Validation Accuracy improves over time
- Training and Validation Loss decreases over time (<0.001 validation loss)