



# Stock Market Prediction with LSTM

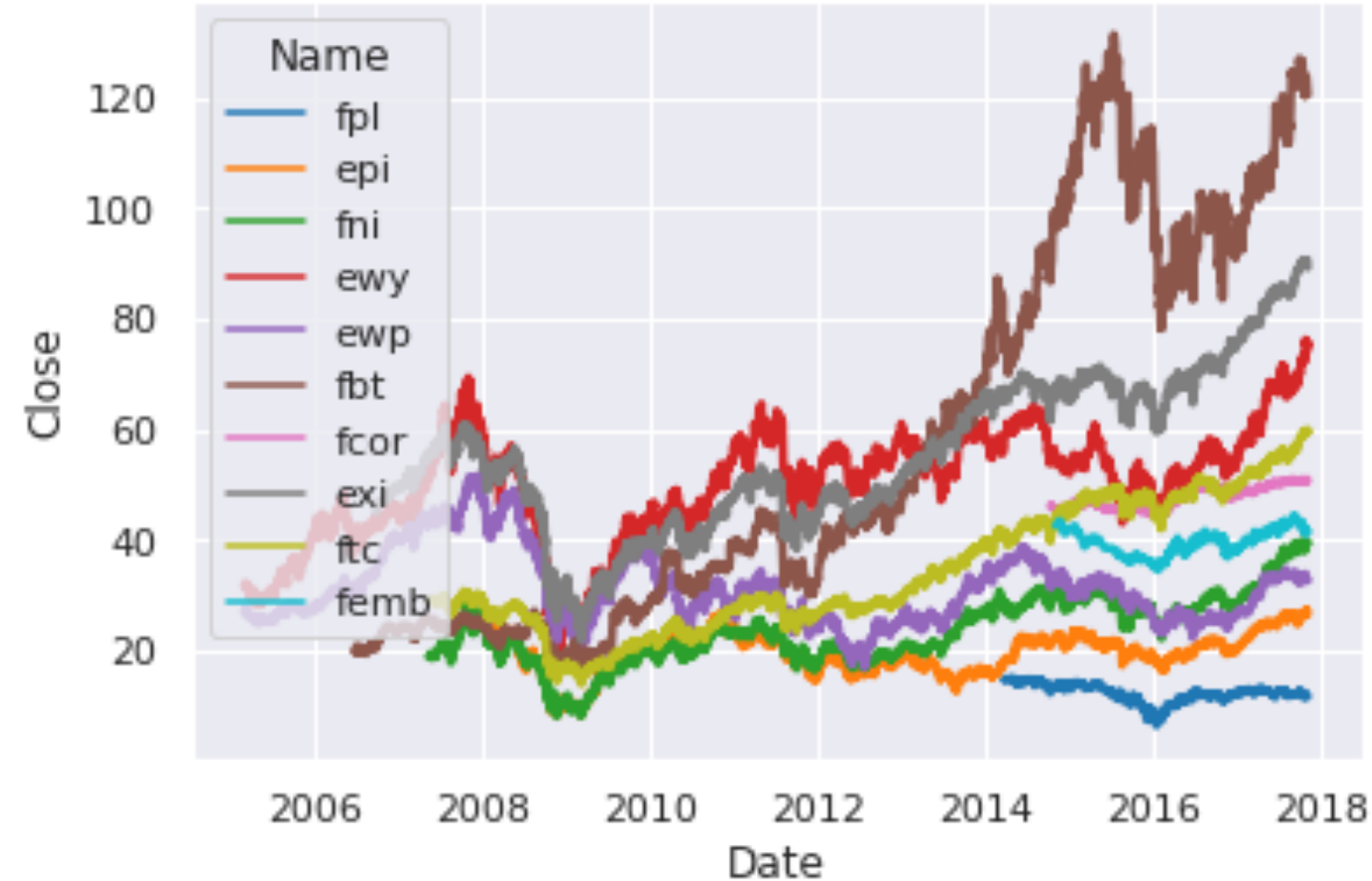
JOSTEIN BARRY-STRAUME

NOVEMBER 27, 2020



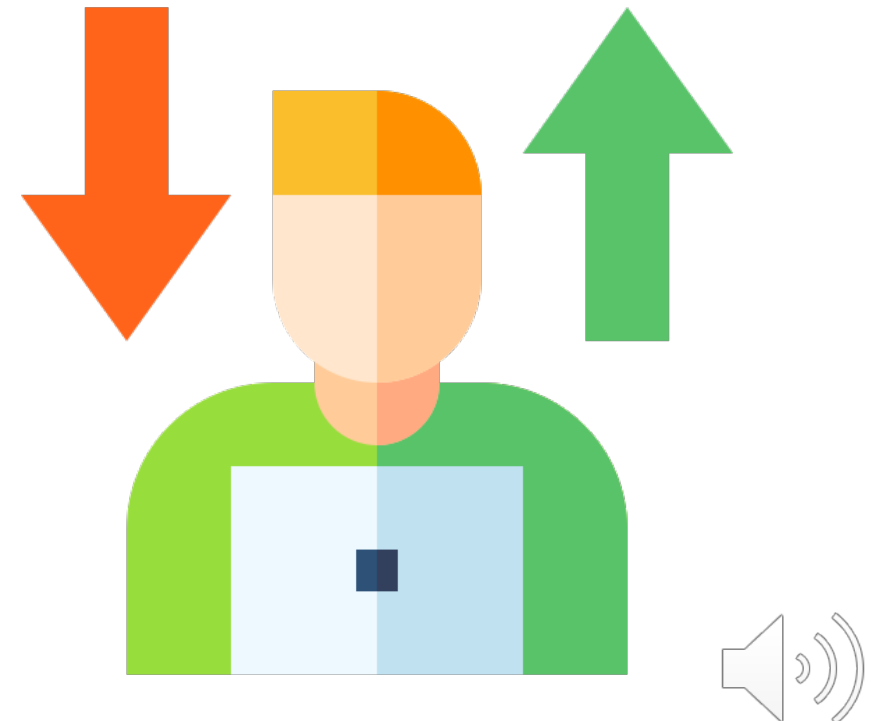
What is the problem?



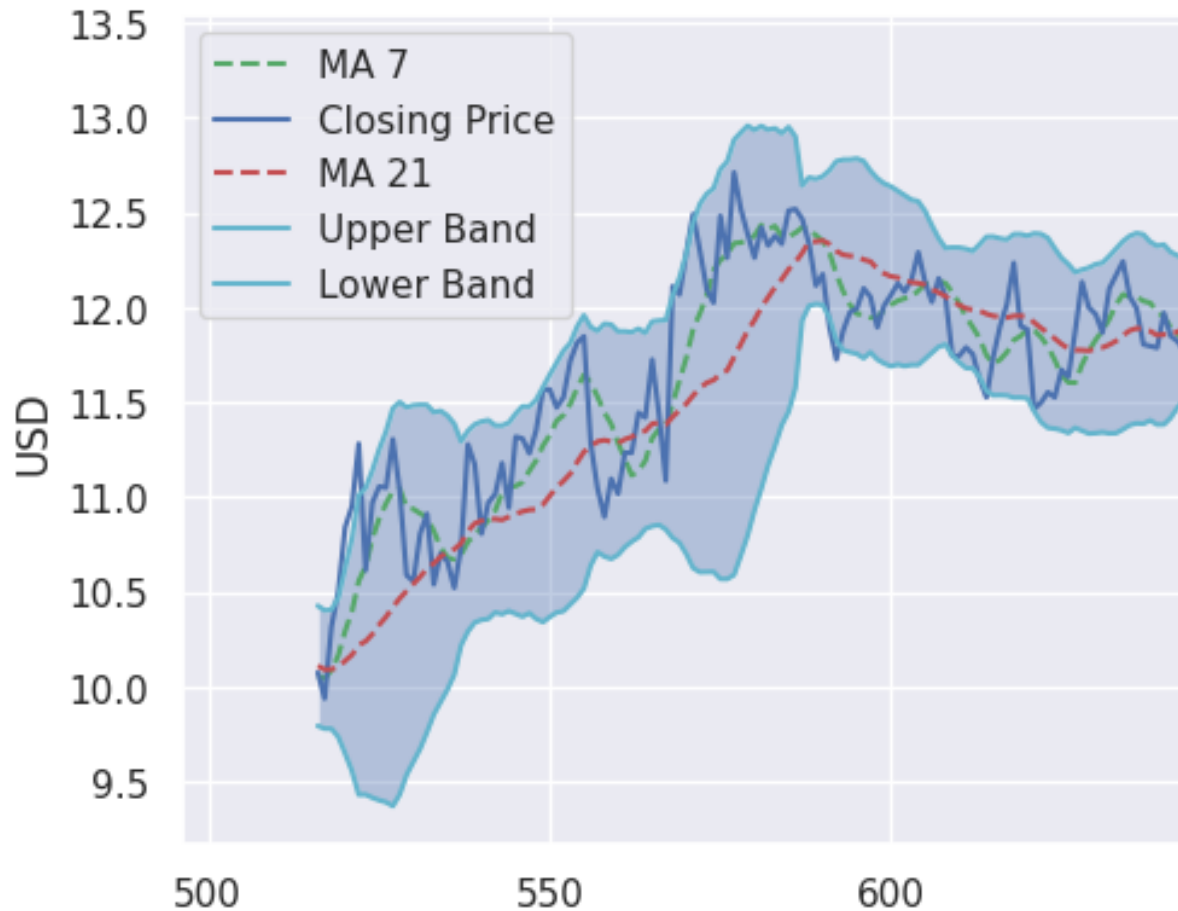


What is the problem?

Train a LSTM neural network to predict if a stock will move up or down



```
# Add the label class based on whether stock goes up or down
def add_label(df):
    idx = len(df.columns)
    new_col = np.where(df['Close'] >= df['Close'].shift(1), 1, 0)
    df.insert(loc=idx, column='Label', value=new_col)
    df = df.fillna(0)
```



What is the problem?

Task:

Supervised learning

Performance Measure:

Validation Accuracy

Learning Component:

Technical Indicators





```
df_etfs.head()
```

|   | Date       | Open   | High   | Low    | Close  | Volume | OpenInt | Name |
|---|------------|--------|--------|--------|--------|--------|---------|------|
| 0 | 2014-03-27 | 14.705 | 14.795 | 14.698 | 14.729 | 698621 | 0       | fpl  |
| 1 | 2014-03-28 | 14.890 | 14.890 | 14.729 | 14.729 | 164979 | 0       | fpl  |
| 2 | 2014-03-31 | 14.839 | 14.948 | 14.729 | 14.876 | 86108  | 0       | fpl  |
| 3 | 2014-04-01 | 14.948 | 14.948 | 14.729 | 14.729 | 169637 | 0       | fpl  |
| 4 | 2014-04-02 | 14.737 | 14.755 | 14.713 | 14.747 | 110332 | 0       | fpl  |

## Data Background

NASDAQ stocks and ETFs

Historical data up until  
04/01/2020

Data Size: 3GB



Why is it an  
important problem?

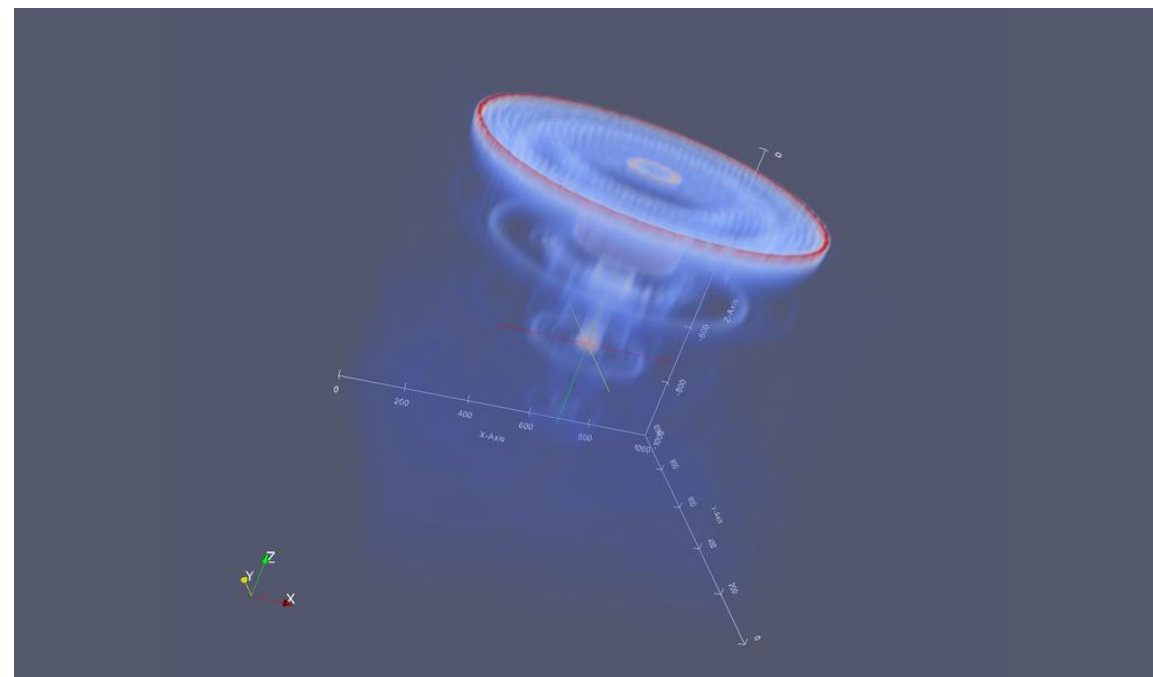






# Motivation

- Lab research
- Career
- Personal interest



# Related work





# Financial time series forecasting model based on CEEMDAN and LSTM



Jia  
Scho

H I

• A  
• T  
• T

A R

Artic  
Rece  
Rece  
Avai

Keyw  
Fina  
EMD  
CEEM



ELSEVIER



CrossMark

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

Science

Procedia Computer S

International Conference on Comp

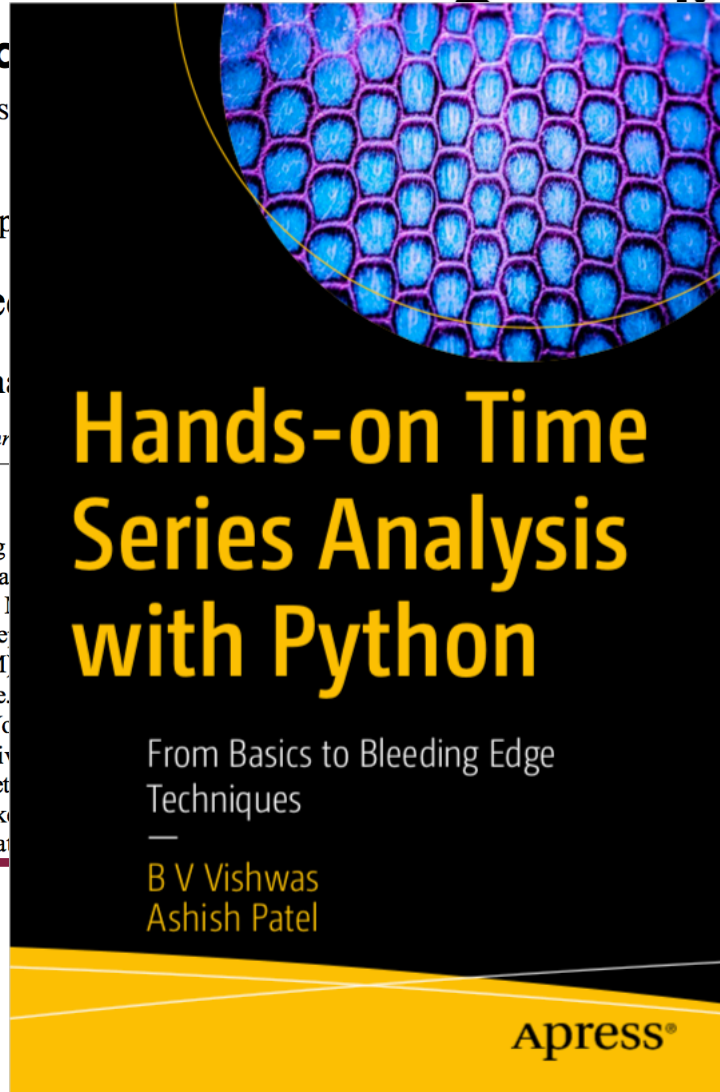
NSE Stock Market Pre

Hiransha M<sup>a</sup>, Gopalakrishna

Centre for Computational Engineering and Networking, Amr

## Abstract

The neural network, one of the intelligent data mining years. Prediction and analysis of stock market data has for forecasting can be categorized into linear (AR, I Network). In this paper, we are using four types of deep Networks (RNN), Long Short-Term Memory (LSTM) of a company based on the historical prices available. National Stock Exchange (NSE) of India and New York of a single company from NSE and predicted for five that CNN is outperforming the other models. The network data. This was possible because both the stock market compared with ARIMA model and it has been observed that



## Literature Review

- Many articles cover implementing hybrid ML models. Not the most helpful for a limited scope class project
- Books, API documentation, and Keras tutorial examples so far have been the most helpful for both understanding and implementing concepts



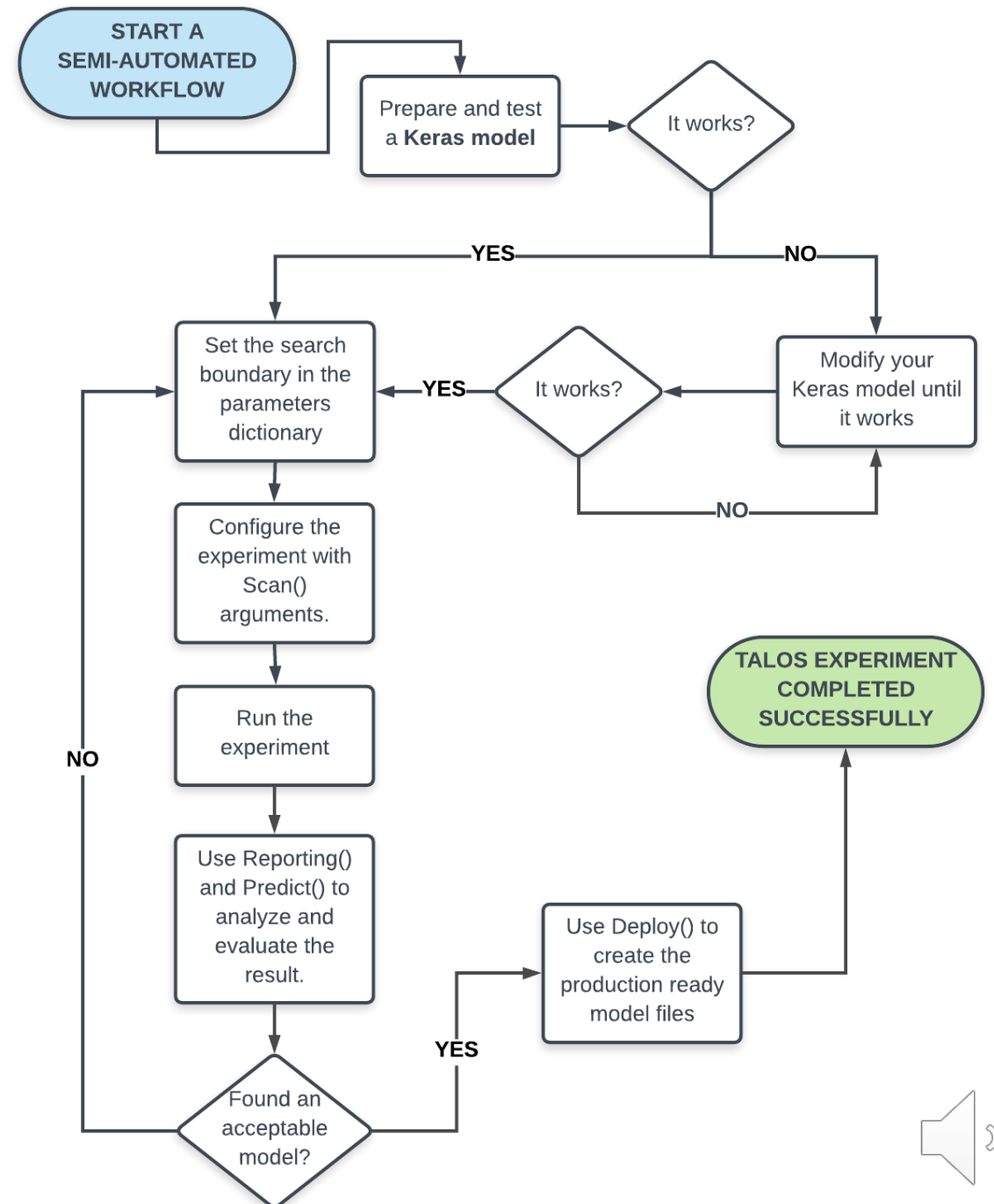
# Techniques chosen and why



## Techniques to tackle problem

- Feature creation
- Data imputation
- Data normalization
- Automated hyper-parameter tuning

```
df_ti = df_ti.fillna(df_ti.median())  
  
def normalized_df(df):  
    normalized_df=(df-df.mean())/df.std()  
    return normalized_df
```



```
p = {'lr': (0.0001, 0.001, 0.005, 0.05, 0.01),  
     'first_neuron': [8, 16, 32, 64],  
     'epochs': [5, 10, 15, 20, 30, 40, 50],  
     'optimizer': ['Adam', 'SGD', 'RMSprop'],  
     'losses': ['binary_crossentropy']}
```

```
def build_lstm(X_train, y_train, X_test, y_test, params):  
    inputs = keras.layers.Input(shape=(X_train.shape[1], X_train.shape[2]))  
    lstm_out = keras.layers.LSTM(params['first_neuron'])(inputs)  
    outputs = keras.layers.Dense(1)(lstm_out)  
    model = keras.Model(inputs=inputs, outputs=outputs)  
  
    model.compile(optimizer=params['optimizer'],  
                  loss=params['losses'],  
                  metrics=['accuracy'])  
    history = model.fit(x=X_train, y=y_train, epochs=params['epochs'], validation_data=(X_test, y_test))  
    return history, model
```

```
%time  
scan_object = ta.Scan(x=X_train,  
                      y=y_train,  
                      model=build_lstm,  
                      params=p,  
                      seed=777,  
                      experiment_name='exp_run_1',  
                      x_val=X_test,  
                      y_val=y_test  
)
```

## Talos API

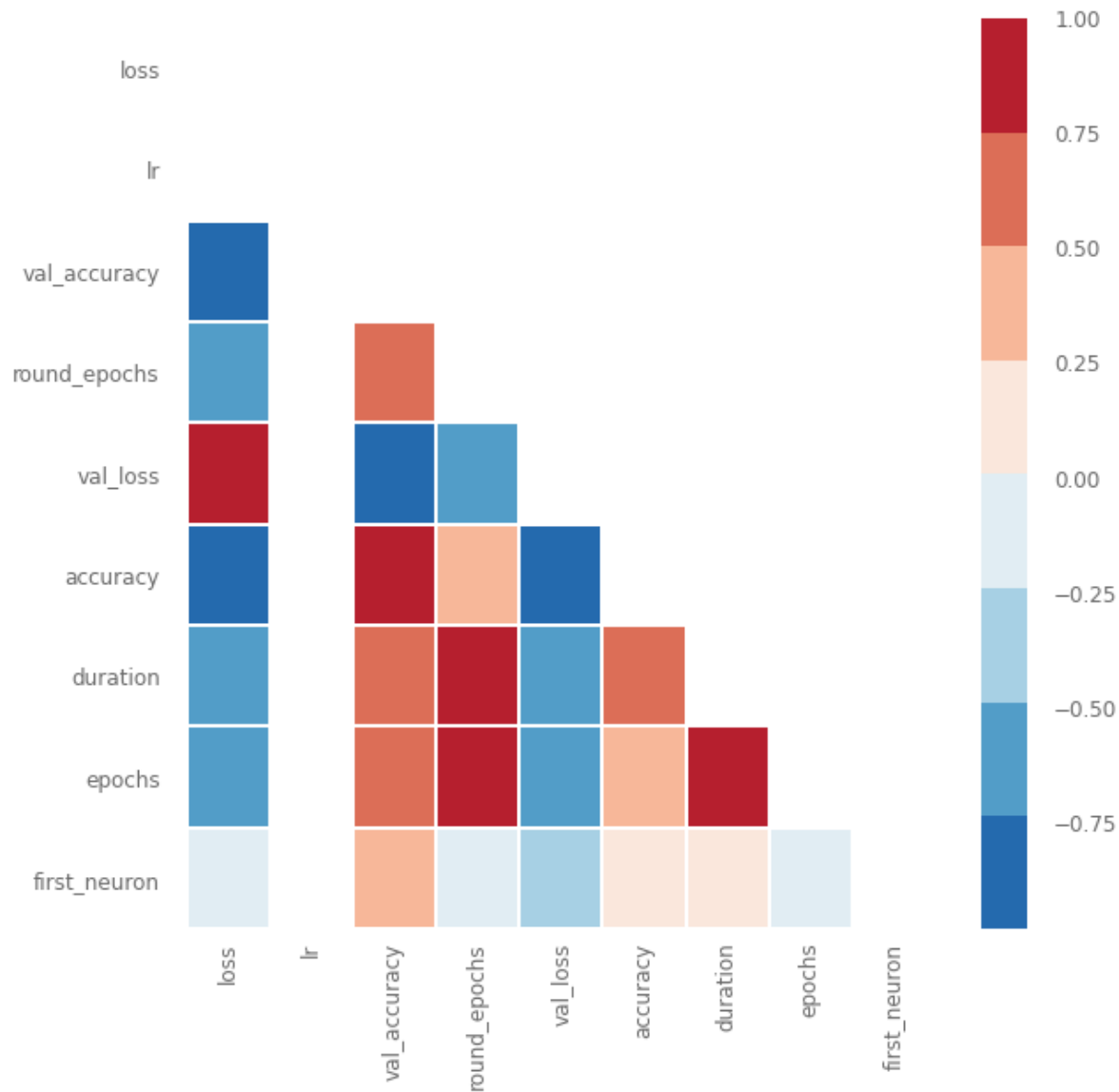
- Set parameters to test
- Build model to specs
- “Scan” all parameter permutations
- “Report” each model’s performance



# Empirical evaluation





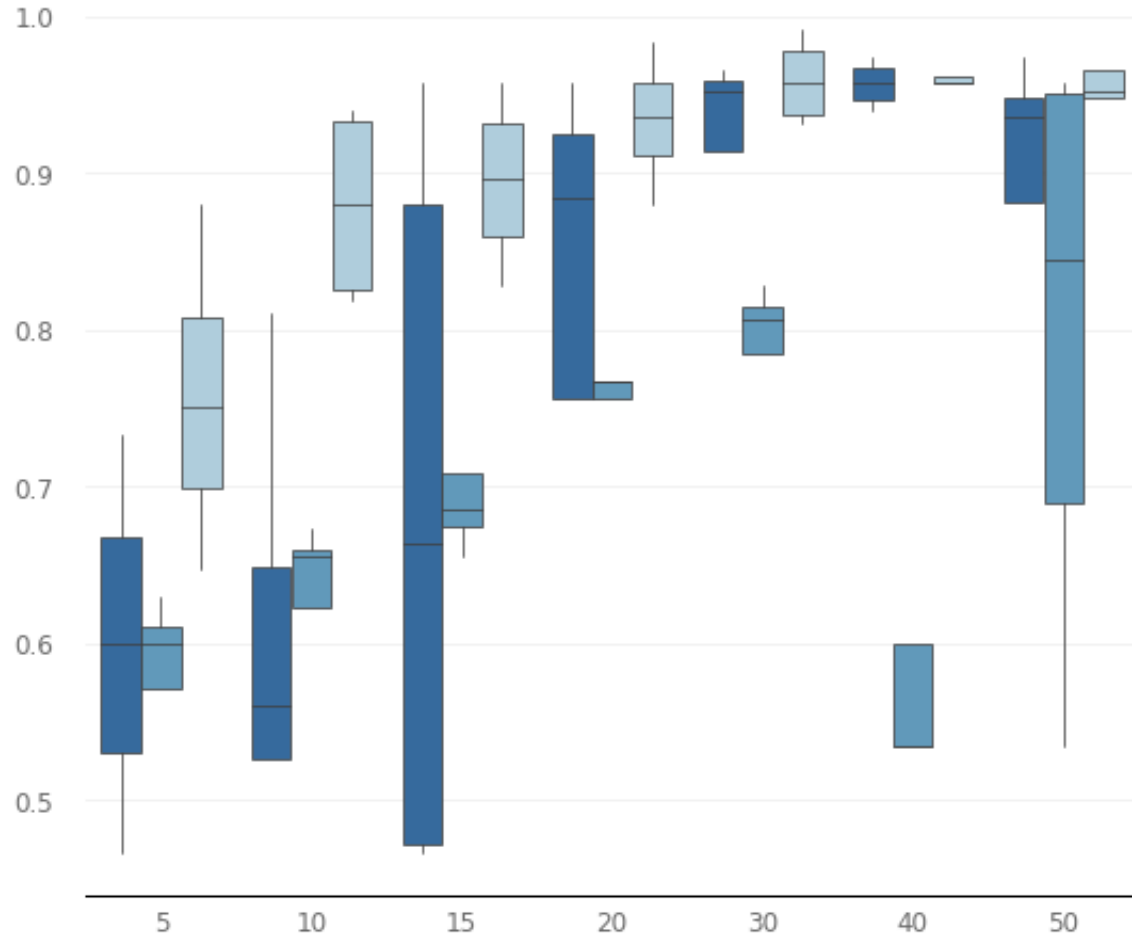


## Visualizing Relationships

Validation Accuracy has strong, positive correlations with:

- Number of neurons
- Epochs
- Duration of training
- Round of epoch training

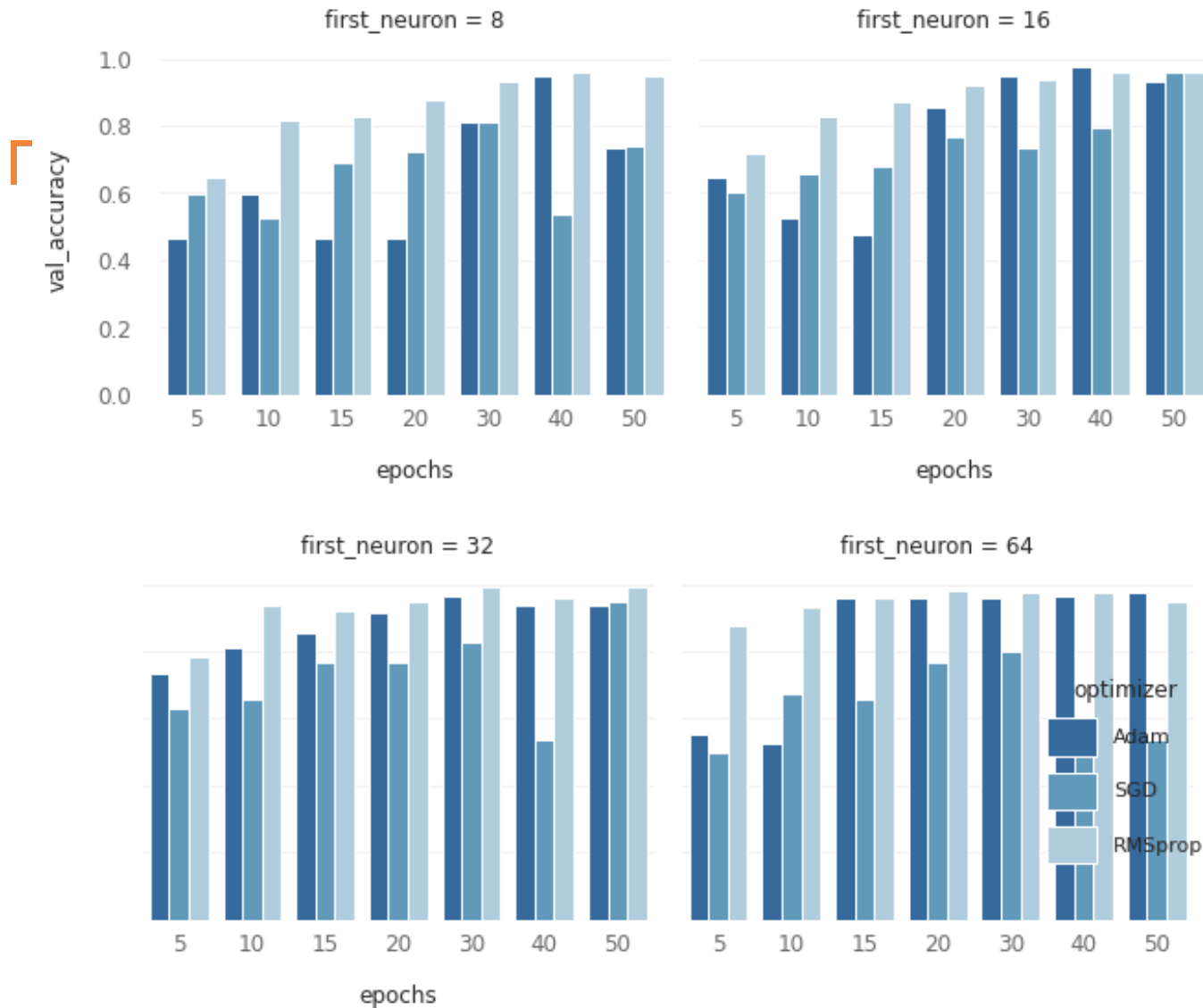




## Validation Accuracy Distribution of Optimizers

- Overall distribution trend is logarithmic
- As epochs increase (x-axis), val acc increases
- Adam has largest variance
- RMSprop has smallest variance





## 4 Dimensional Bar Grid

Breaks down validation accuracy for each neuron density by epochs per each optimizer

- RMSprop performs consistently better in each neuron density
- SGD seems to struggle at highest neural density; perhaps overfitting?



# Conclusion



```
# returns the highest value for validation accuracy
analyze_object.high('val_accuracy')
```

```
0.9913793206214905
```

```
# returns the number of rounds it took to find best model
analyze_object.rounds2high('val_accuracy')
```

```
56
```

## Best Model

- 84 different models trained
- Best model:
- Found at round 56 of 84
- Validation accuracy: 99.14%

| Round | Val Acc | LR     | Epochs | Optimizer | Neural Density |
|-------|---------|--------|--------|-----------|----------------|
| 56    | 99.14%  | 0.0001 | 50     | RMSprop   | 32             |





# Questions?





**VIRGINIA TECH™**

