RNN Stock Predictions

Predicting stock market returns is field that has been tried many times before. For my final project I have attempted to do the same.

Problem Statement

- One large segment of the financial sector focuses on finding good investment opportunities.
- This part of the industry still employs methods that were developed before modern data science practices and tools were created
- There is a large opportunity to modernize this segment of the financial industry.
- A person that can analyze more companies has an advantage over other investment professionals

Related Work

- Many of the top hedge funds currently employ quantitative methods to solve for this problem.
- Many of the top investment banks employee ”quants” or “strats” to solve for this problem.
- Recently there has been a rise in AI investing which has allowed people to access investment opportunities driven by data mining methods
- There are many guides to the type of analysis that should be done on sites like Investopedia.

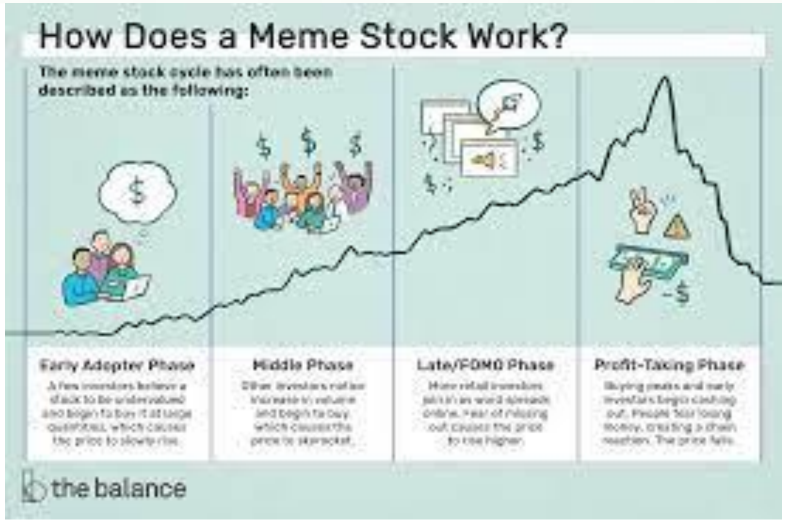
Proposed Data

- Technical Analysis
  - Traders believe in the utilization of many different types of technical analysis. Some examples are the comparison of current stock prices to the moving average to determine if the stock is over bough or over sold.
- Fundamental Analysis
  - Analysts look at company financial documents to understand if there is value in the company that currently isn’t being priced into the stock price. Much of this data is available through APIs. I would like to study the stock price of many different companies compared to their underlying metrics to see if fundamental analysis works.
- Event Analysis
  - With the rise of meme stocks a company’s stock price may increase regardless of the economic, market, or fundamental situation. I would like to study the impact of the sentimentality of the news on a stock price.
- Economic Analysis
  - The condition of the wider economy plays a large role in the success of individual companies. Generally when the economy is not doing well a company will have a harder time generating wealth.


Springfield Psychological Services

Balance Sheets
December 31, 2004 and 2003

	2004	2003		2004	2003
Assets			**Liabilities**		
Current assets:			Current liabilities:		
Cash	$ 12,597	$ 8,173	Note payable	$ 4,200	$ 4,752
Short-term investments	5,000	3,517	Accounts payable	375	15
Accounts receivable	2,315	3,750	Accrued wages	1,579	1,149
Prepaid Rent	3,000	3,000	Taxes payable	5,386	4,722
Total current assets	22,915	18,440	**Total current liabilities**	11,540	10,638
Property, plant and equipment					
Land and building	65,553	28,369	**Long-term debt**	50,000	20,000
Machinery and equipment	5,000	3,511			
	70,553	31,880	**Owner's Equity**		
Less accumulated depreciation	5,775	4,321	**Total owners' equity**	27,789	28,159
Property and equipment, net	64,778	27,558			
Long-term investments	1,353	4,587	**Total liabilities and**		
Other assets	283	211	**owners' equity**	$ 89,329	$ 58,797
Total assets	$ 89,329	$ 58,797			

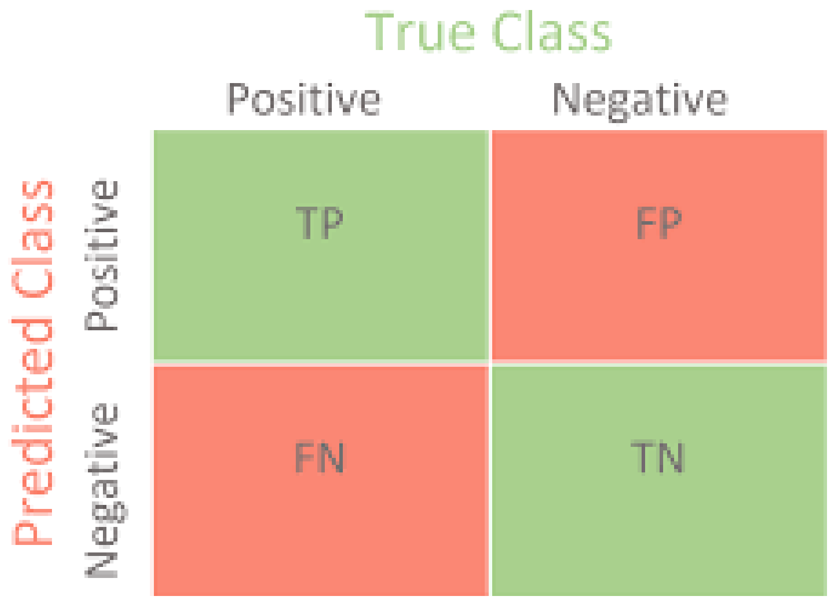


Proposed Model

- For this project I have chosen to develop an RNN model. Specifically I will be using GRU because of its efficiency and accuracy.
- I will be using keras_tuner to tune the hyperparameters of the algorithm to hopefully get the best version of my model
- I will be using Keras as my model framework to develop the dataset and the model.

Evaluation

- I will evaluate my model based on the accuracy of the predictions.
  - Since I am using a regression I will be minimizing the L1 distance between the prediction and the true value of the regression.



Timeline

Data Engineering → **Data Mining** → **Data Modeling** → **Evaluation**

Data Engineering

- Identify good data sources.
- Produce a dataset rich with features
- Produce a dataset rich with labels

Data Mining

- Perform basic EDA on the dataset
- Identify most promising features

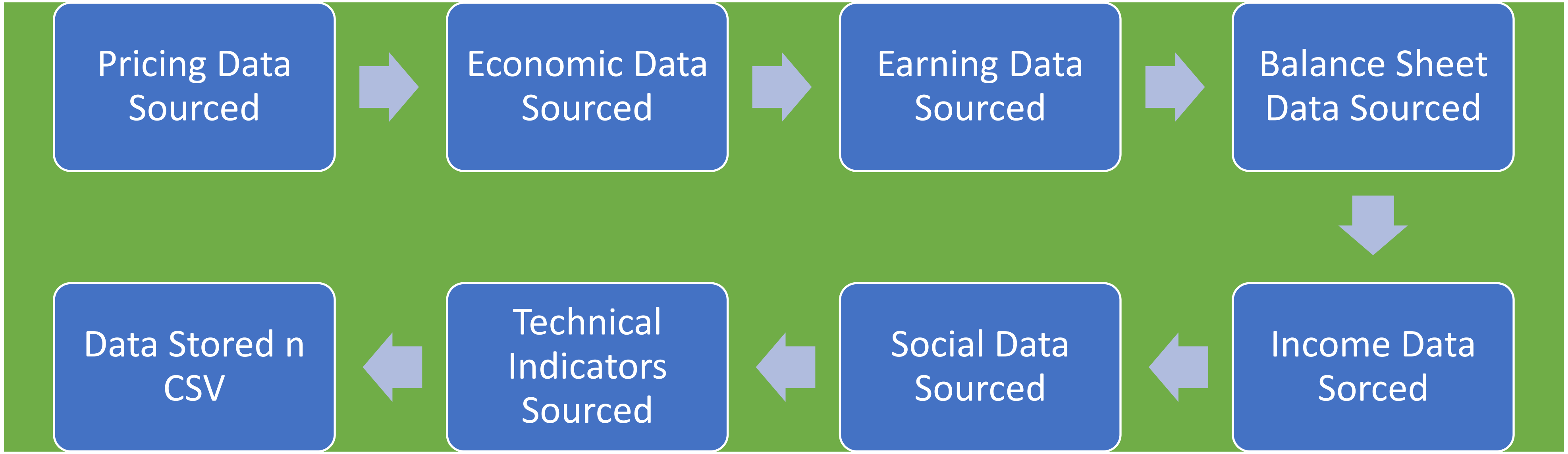
Data Modeling

- Build Models to make predictions
- Evaluate Accuracy of predictions

Evaluation

- Summarize Analysis
- Finish final project

Data Engineering Work Completed



The necessary data has been sourced for this project.

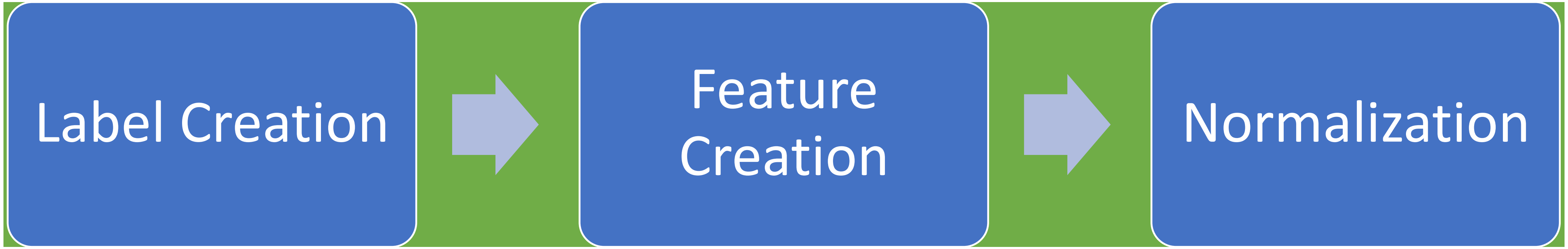
After upgrading to a computer with more resources all the data could be sourced into memory in a single flat file.

Size of Data ~4GB

Number of data points in the train set: 3,889,623, number of used features: 102

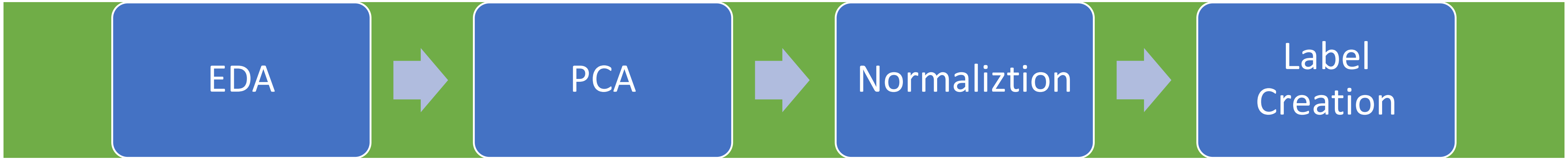
The dataset now encompasses all companies in the NASDAQ

Feature Engineering



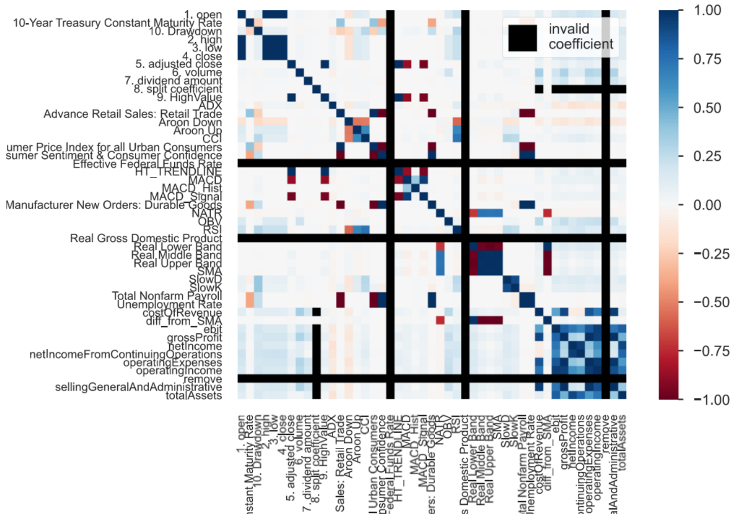
- Label Creation
    - Forward Price Prediction:
        - Labels have been created for training a 1,2,7,30,90,365 day prediction
        - Labels have been created for training a binary up down prediction
- Feature Creation
    - Features are being created to add domain knowledge to the algorithm.
        - Comparison to moving averages
        - Comparison to other indicators
        - Fields to help algorithm understand old vs new datapoints.
- Normalization
    - Normalization using Standard normalization has been completed
- PCA
    - The dataset has grown to over a 100 features. I have used PCA to narrow that set of features to 30 which explain 90% of the variance.

Data Mining Work Completed

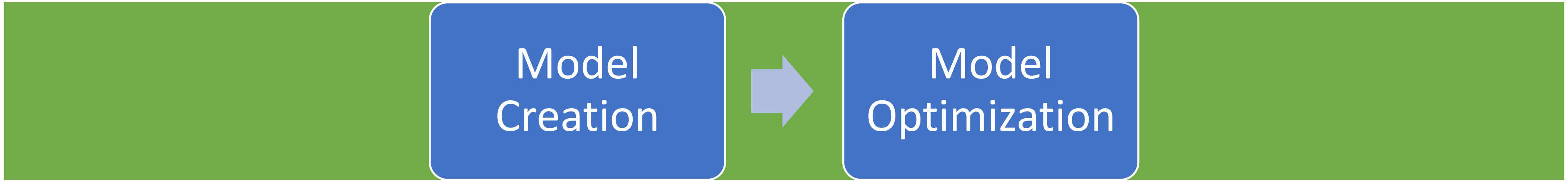


- EDA
  - Using pandas profiling a python package basic EDA has been completed to judge the usability of the 100 + fields.
  - Correlation as well as studies on missing data have led ton increased accuracy
- PCA
  - A PCA analysis has been completed to identify variance attribution in the data
  - Dimensionality reduction may be conducted as the number of feature grow

Feature	PCA	Abs PCA
AroonUp	-0.77	0.77
AroonDown	0.48	0.48
SlowK	-0.25	0.25
SlowD	-0.24	0.24
RSI	-0.21	0.21
CCI	-0.07	0.07
ADX	-0.05	0.05
10Drawdown	-0.04	0.04
10YearTreasuryConstantMaturityRate	0.02	0.02
4close	-0.01	0.01
3low	-0.01	0.01
2high	-0.01	0.01
1open	-0.01	0.01
OBV	0.00	0.00



Data Modeling



- RNN - GRU
  - The main model being utilized is GRU. To the right is the model architecture after the last optimization round.
- Optimization
  - All models are being optimized using python framework keras_tuner to protect against overfitting and to ensure optimal model performance.

```
Search: Running Trial #215

Value             |Best Value So Far |Hyperparameter
25                |5                 |output_dm
100               |10                |input_unit
0                 |0                 |Dropout_rate
sigmoid           |relu              |dense_activation
12                |34                |tuner/epochs
0                 |12                |tuner/initial_epoch
2                 |4                 |tuner/bracket
0                 |3                 |tuner/round

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 25)          750

 gru (GRU)                   (None, None, 100)         38100

 dropout (Dropout)           (None, None, 100)         0

 dense (Dense)               (None, None, 1)           101

=================================================================
Total params: 38,951
Trainable params: 38,951
Non-trainable params: 0
```

Evaluation

Profitability Measure → MAE Model Evaluation

- Profitability is the most import measure for a model like this. We will compare the model output to the real data to determine profitability if it were deployed

Row Labels	Sum of Money Made
2017	$ 96.67
Qtr3	$ 59.07
Qtr4	$ 37.60
2018	$ (110.00)
Qtr1	$ (70.22)
Qtr2	$ 41.39
Qtr3	$ (50.60)
Qtr4	$ (30.57)
2019	$ 383.24
Qtr1	$ 68.14
Qtr2	$ 63.87
Qtr3	$ 102.07
Qtr4	$ 149.16
2020	$ 176.59
Qtr1	$ (110.03)
Qtr2	$ 185.04
Qtr3	$ (4.82)
Qtr4	$ 106.41
2021	$ 591.28
Qtr1	$ 127.50
Qtr2	$ 115.51
Qtr3	$ 375.02
Qtr4	$ (26.75)
2022	$ (174.75)
Qtr1	$ (166.91)
Qtr2	$ (9.38)
Qtr3	$ 1.53
Grand Total	$ 963.03

```
Best val_mean_absolute_error So Far: 0.12859368324279785
Total elapsed time: 02h 54m 46s

Search: Running Trial #214

Value             |Best Value So Far |Hyperparameter
10                |5                 |output_dm
140               |10                |input_unit
0.2               |0                 |Dropout_rate
relu              |relu              |dense_activation
12                |34                |tuner/epochs
0                 |12                |tuner/initial_epoch
2                 |4                 |tuner/bracket
0                 |3                 |tuner/round

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 10)          300

 gru (GRU)                   (None, None, 140)         63840

 dropout (Dropout)           (None, None, 140)         0

 dense (Dense)               (None, None, 1)           141

=================================================================
Total params: 64,281
Trainable params: 64,281
Non-trainable params: 0
_________________________________________________________________
Epoch 1/12
1880/1880 [==============================] - 15s 8ms/step - loss: 0.1291 - mean_absolute_error: 0.1291 - val_loss: 0.1286 - val_mean_absolute_error: 0.1286
Epoch 2/12
1012/1880 [==============>...............] - ETA: 6s - loss: 0.1288 - mean_absolute_error: 0.1288
```

Code Snapshots

Model → PCA → Normalization → Optimization

```python
def pca_transform(self):
    print(np.any(np.isnan(self.X_valid)))
    self.X_train = np.nan_to_num(self.X_train, copy=True, nan=0.0, posinf=None, neginf=None)
    self.X_valid = np.nan_to_num(self.X_valid, copy=True, nan=0.0, posinf=None, neginf=None)
    print(np.any(np.isnan(self.X_valid)))
    #self.pca = PCA(.90, svd_solver='full')
    self.pca = PCA(n_components = 30, svd_solver='full')
    self.pca.fit(self.X_train)

    print(self.pca.n_features_)
    print(self.pca.n_samples_)
    print(self.pca.components_)
    print(self.pca.explained_variance_)
    print(self.pca.explained_variance_ratio_)

    self.X_train_pca = self.pca.transform(self.X_train)
    self.X_valid_pca = self.pca.transform(self.X_valid)

    print(len(self.X_train_pca[0]))
```

```python
def normalization(self, data, scale):
    if scale == 'MinMax':
        self.scaler = MinMaxScaler()
    elif scale == 'Standard':
        self.scaler = StandardScaler()
    self.scaler.fit(data)
    dump(self.scaler, open('model/scaler.pkl', 'wb'))

def label_scaler(self, data):
    y = np.array(data)
    y = y.reshape(-1, 1)
    self.scaler = MinMaxScaler()
    y = self.scaler.fit_transform(y)
    y = y.reshape(1, -1)[0]
    return y
```

```python
def train_algorithm(self):
    epocheroos = 100
    tuner = kt.Hyperband(
        hypermodel=self.build_model,
        objective= kt.Objective(name="val_mean_absolute_error",direction="min"),
        max_epochs=epocheroos,
        factor=3,
        hyperband_iterations=1,
        directory='neural_optimizer',
        project_name='hyperband',
        overwrite= False
    )
    stop_early = EarlyStopping(monitor='val_mean_absolute_error', patience=5)
    print("Starting Tuner Search")
    tuner.search(self.X_train_pca, self.y_train, epochs=epocheroos, validation_split=0.2, callbacks=[stop_early])

    # # Get the optimal hyperparameters
    print("Starting Get Best Hyperparameters")
    best_hps = tuner.get_best_hyperparameters(num_trials=3)[0]

    # Build the model with the optimal hyperparameters and train it on the data for x epochs
    print("Building model with Best Hyperparameters")
    model = tuner.hypermodel.build(best_hps)
    history = model.fit(self.X_train_pca, self.y_train, epochs=epocheroos, validation_split=0.2)

    print("Getting Best Epoch from the history")
    val_acc_per_epoch = history.history['val_mean_absolute_error']
    best_epoch = val_acc_per_epoch.index(max(val_acc_per_epoch)) + 1
    print('Best epoch: %d' % (best_epoch,))

    hypermodel = tuner.hypermodel.build(best_hps)

    # Retrain the model
    print("Retraining with best Epoch")
    hypermodel.fit(self.X_train_pca, self.y_train, epochs=best_epoch, validation_split=0.2)

    print("Starting Eval of best Model")
    eval_result = hypermodel.evaluate(self.X_valid_pca, self.y_valid)
    print("[test loss, test accuracy]:", eval_result)
    hypermodel.save(f'model/neural_net_model_01_{days}_days.pkl')
    self.model = hypermodel
```

```python
def build_model(self, hp):
    model = Sequential()
    model.add(Embedding(input_dim=len(self.X_train_pca[0]), output_dim=hp.Int('output_dm',min_value=5,max_value=50,step=5))),
    model.add(GRU(hp.Int('input_unit',min_value=10,max_value=200,step=10),return_sequences=True))
    # for i in range(hp.Int('n_layers', 1, 4)):
    #     model.add(LSTM(hp.Int(f'lstm_{i}_units',min_value=32,max_value=512,step=32),return_sequences=True))
    # model.add(LSTM(hp.Int('layer_2_neurons',min_value=32,max_value=512,step=32)))
    model.add(Dropout(hp.Float('Dropout_rate',min_value=0,max_value=0.5,step=0.1)))
    model.add(Dense(1, activation=hp.Choice('dense_activation',values=['relu', 'sigmoid'],default='relu')))
    model.compile(loss=MeanAbsoluteError(),
                  optimizer=Adam(1e-4),
                  metrics=MeanAbsoluteError())
    model.summary()
    return model
```