CHALLENGES OF STOCK PRICE PREDICTIONS USING RNNS

ILYA BULYGIN

PROBLEM STATEMENT

- Goal of the project was to **analyze different parameters** which affect the accuracy of a Recurrent Neural Network (RNN) predicting the stock prices, **derive meaningful conclusions** from the results of the tests and **assess the quality** of the resultant fine-tuned model
- Research provides **detailed evaluation** of the results related to difficulties and specifics of fine-tuning such a network as well as **applications of the results to the real life**

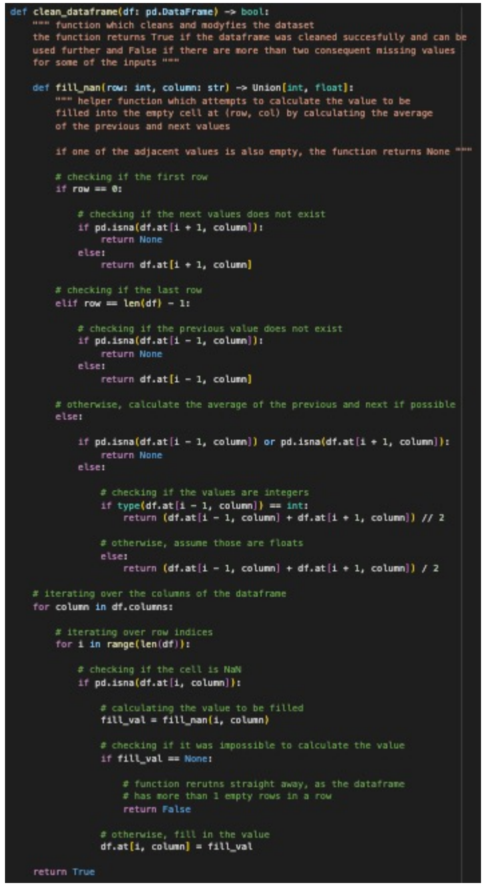
METHODS

```
model = Sequential()
model.add(SimpleRNN(
            units = 140,
            activation = 'relu',
            input_shape = (X_train.shape[1], X_train.shape[2])))
model.add(Dense(units = 1))

model.compile(
            optimizer = 'adam',
            loss = 'mean_squared_error')
```

- **Data Source**: Stock Market Dataset from Kaggle
  - 5885 stocks
  - 2.34 GB of data
- **Model Structure**: two-layer neural network
  - RNN layer with 140 units, RELU activation
  - Dense output layer with 1 unit
- **Model Compilation**:
  - Adam optimizer
  - Mean Squared Error loss

CHALLENGE

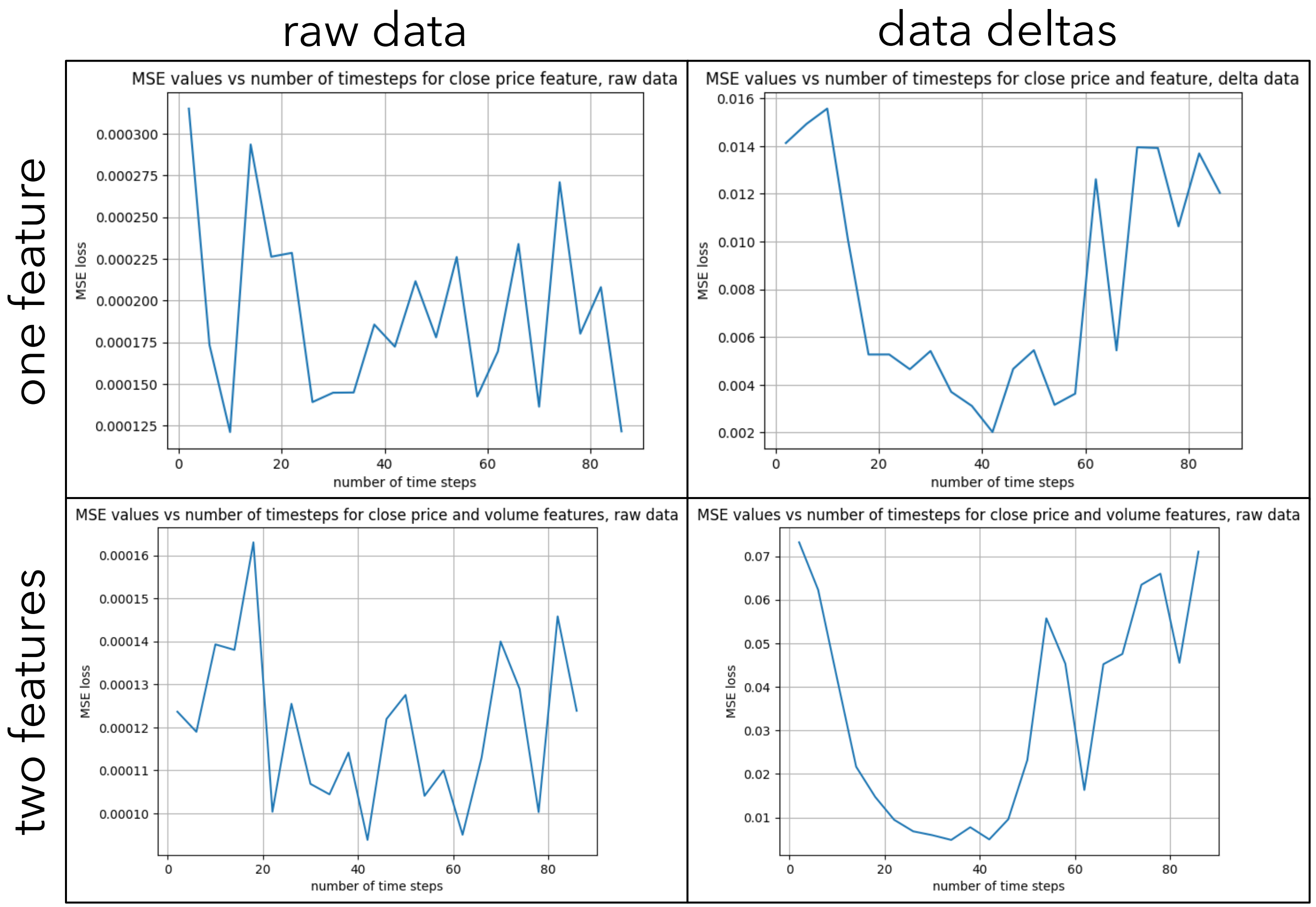
- **Dataset Cleaning**: dealing with missing data, which started causing crashes of the model
- **Optimizing Code for Google Colab Crashes**: rewriting the code in a way to avoid result loss due to Google Colab crashes by dividing it into sections and cells to be executed together, without affecting others.
- **Balancing Between Accuracy and Time Efficiency**: finding the appropriate number of epochs for model training and step for adjusted parameter when comparing models trained for different parameters.



```python
def clean_dataframe(df: pd.DataFrame) -> bool:
    """ function which cleans and modyfies the dataset
    the function returns True if the dataframe was cleaned succesfully and can be
    used further and False if there are more than two consequent missing values
    for some of the inputs """

    def fill_nan(row: int, column: str) -> Union[int, float]:
        """ helper function which attempts to calculate the value to be
        filled into the empty cell at (row, col) by calculating the average
        of the previous and next values

        if one of the adjacent values is also empty, the function returns None """

        # checking if the first row
        if row == 0:

            # checking if the next values does not exist
            if pd.isna(df.at[i + 1, column]):
                return None
            else:
                return df.at[i + 1, column]

        # checking if the last row
        elif row == len(df) - 1:

            # checking if the previous value does not exist
            if pd.isna(df.at[i - 1, column]):
                return None
            else:
                return df.at[i - 1, column]

        # otherwise, calculate the average of the previous and next if possible
        else:

            if pd.isna(df.at[i - 1, column]) or pd.isna(df.at[i + 1, column]):
                return None
            else:

                # checking if the values are integers
                if type(df.at[i - 1, column]) == int:
                    return (df.at[i - 1, column] + df.at[i + 1, column]) // 2

                # otherwise, assume those are floats
                else:
                    return (df.at[i - 1, column] + df.at[i + 1, column]) / 2

    # iterating over the columns of the dataframe
    for column in df.columns:

        # iterating over row indices
        for i in range(len(df)):

            # checking if the cell is NaN
            if pd.isna(df.at[i, column]):

                # calculating the value to be filled
                fill_val = fill_nan(i, column)

                # checking if it was impossible to calculate the value
                if fill_val == None:

                    # function rerutns straight away, as the dataframe
                    # has more than 1 empty rows in a row
                    return False

                # otherwise, fill in the value
                df.at[i, column] = fill_val

    return True
```

RESULTS: NUMBER OF STEPS

- Identifying an **optimal number of steps** and the format of fed data
- Data deltas performed worse, making it reasonable to make a conclusion about **information lost from absolute values**
- Performance peak for **approximately 40 days**, most affectionate period

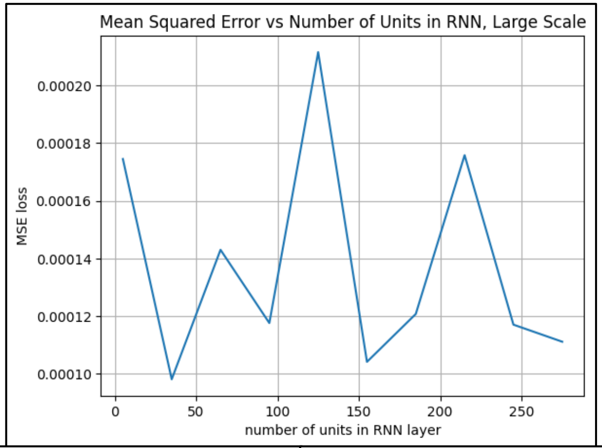


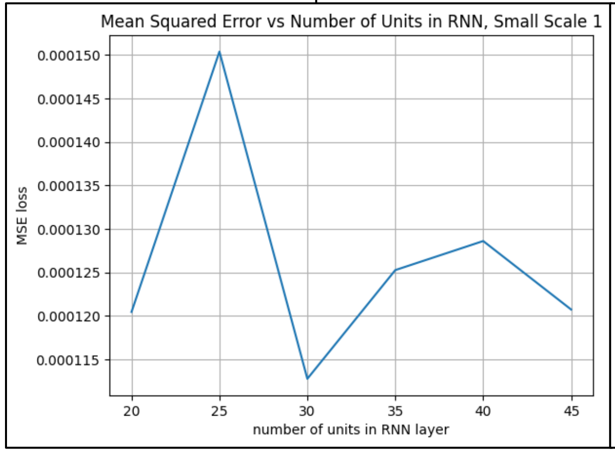
RESULTS: BEST FEATURE SET

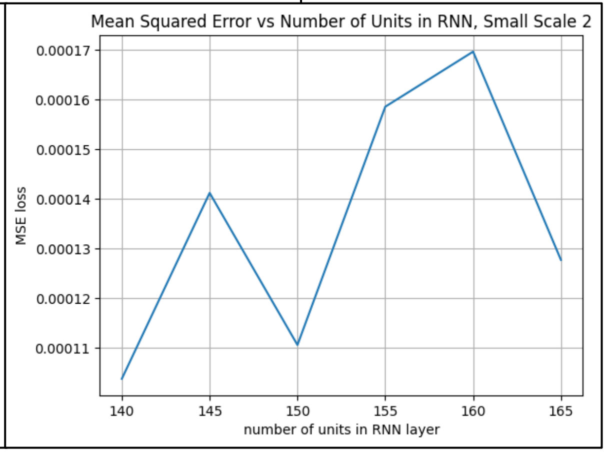
```
[(0.00010678591934265569, 'Close, Adj Close'), (0.00011345745588187128, 'Close, Volume'), (0.0001236729440279305, 'Close, Open')]
[(8.826652629068121e-05, 'Close, Adj Close, Volume'), (9.846078319242224e-05, 'Close, High, Low'), (0.00010334944818168879, 'Close, High, Adj Close')]
[(0.00015594298020005226, 'Close, High, Adj Close, Volume'), (0.00015776506916154176, 'Close, Low, Adj Close, Volume'), (0.00021284236572682858, 'Close, High, Low, Volume')]
[(0.00019055783923249692, 'Close, High, Low, Adj Close, Volume')]
```

- Counterintuitively, **using all possible features for training does not result in the best performance**, as smallest loss appears in networks, trained on 3 features
- **Volume data** appearing in the most effective feature sets for all number of features, making it reasonable to conclude about the impact of daily trade volume on price prediction
- **Open price** not appearing in any of the best-performing feature sets, making it reasonable to conclude about irrelevance on this data, as it reflects on overnight or weekend rumors
- Conclusion about data repetitiveness in the context of stock prices, as **most metrics repeat themselves**, causing the model to **overfit**

RESULTS: NUMBER OF RNN UNITS







- On a large scale best results are produced for **around 40 and around 150 unit ranges**
- Conclusion about **more complex correlation** found using 140 units
- Slightly **better performance for 140 units** according to the results of smaller scales
- With more training, **140-unit network is more ambitious**

RESULTS: NETWORK PERFORMANCE

- Mean Squared Error **losses on the training and testing sets are similar**, of an order of 10^-4
- **Directional accuracy of price predictions of 87%**, making the model ambitious in terms of combining in with and efficient trading strategy
- Implemented **simple trading strategy**
  - If price goes up, buy 1 unit of stock
  - If price goes down, sell all stocks available
- Resultant returns are relatively low, yet **all positive**, making it reasonable to claim the model is **safe for trading** and can bring greater revenues if used with appropriate trading strategy

```
Epoch 100/100
464/464 [==============================] - 14s 30ms/step - loss: 1.1032e-04
```

```
116/116 [==============================] - 1s 5ms/step - loss: 1.1354e-04
Mean Squared Error on the Testing Set is  0.00011353728041285649
```

```
dir_accuracy = get_directional_accuracy(model_final, [close_vals, adj_close_vals, volume_vals])
print("Directional Accuracy of the Model is " + str(round(100 * dir_accuracy, 2)) + "%")

Directional Accuracy of the Model is 87.29%
```

```
Stock: CBSHP
Return On Investment: 11.1%
Annual Return: 2.74%
Annual Profit: 9.55$

Stock: CINR
Return On Investment: 11.11%
Annual Return: 2.42%
Annual Profit: 10.27$

Stock: BKTI
Return On Investment: 30.71%
Annual Return: 0.98%
Annual Profit: 4.11$
```

```
Return On Investment for Microsoft: 9.75%
Return On Investment for Amazon: 14.98%
Return On Investment for Tesla: 28.92%
```

```
Annual Return for Microsoft: 0.4%
Annual Return for Amazon: 0.9%
Annual Return for Tesla: 3.91%
```

```
Annual Profit for Microsoft: 2.8$
Annual Profit for Amazon: 5.57$
Annual Profit for Tesla: 10.05$
```

CONCLUSIONS

- As the model is more accurate when the predictions are based on raw data (absolute price of the stock) instead of deltas, which means that **larger companies with more expensive stocks behave differently on the financial market compared to smaller ones**
- Most fluctuations in the stock market are affected by the data of the **40 latest days**, meaning means that in most cases **financial reports outside of the 40 days window barely affect price fluctuations in the future**
- **Open price records are mostly misleading for stock price predictions**, as adding them to the training data worsens the results
- Stock market has **numerous levels of complexity of data correlation**, confirmed by peaks of performance of models with specific numbers of units in RNN

FUN FACT

IN THE U.S. EQUITY MARKET, EUROPEAN FINANCIAL MARKETS, AND MAJOR ASIAN CAPITAL MARKETS, ALGORITHMIC TRADING ACCOUNTS FOR ABOUT 60-75 PERCENT OF THE OVERALL TRADING VOLUME

THANK YOU
FOR YOUR
ATTENTION!