Task 2: Time Series Forecasting using Genetic Algorithm

Zeynep Öykü Erdem, 100002305

July 9, 2025

Contents

1 Time Series Forecasting

Time Series Forecasting is a serious of collected, measured data points at evenly spaced time intervals. According to the past observations, we predict the future values.

2 Problem Definition

The Objective of this project to obtain better generation of solutions for time series forecasting using generic algorithm. The model aims to predict the next day's adjusted close of price of Netflix stock .

3 Dataset

Netflix Stock Price Dataset, consists of 6 columns , as follows Date, Open, High,Low, Close, Adj Close and Volume. Open column specifies the opening value of the stock current day, High and Low columns specify the high and low value of the stock in the current day, Close column specifies the closed value ,and Adj Close column adjusts the Close price for corporate actions such as :

- Stock splits
- Dividends
- Rights offerings

Volume column expresses the number of shares traded.

Dataset consists of 1009 entries :

#	Column	Dtype
0	Date	object
1	Open	float64
2	High	float64
3	Low	float64
4	Close	float64
5	Adj Close	float64
6	Volume	int64

Table 1: Data Types

3.1 Data Visualization before Normalization

In this project, the aim is to find the adjusted close value the next day by using a sequence of 60. First from kagglehub I downloaded the csv, then by using pandas library I read the csv file and stored the data in data variable.

Because we are using a large scale of values in different features, I used MinMaxScaler to normalize the data in order to obtain stable range of values,better performance, less memory usage.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

Figure 1: First 5 rows of the dataset

```
def create_sequences(features,target,sequence_length):
  X,y=[],[]
  for i in range(len(features) - sequence_length-1):
        X.append(features[i:i+sequence_length])
        y.append(target[i+sequence_length])
  X,y=np.array(X),np.array(y)
  return X,y
```

The create_sequences function takes 2 parameters in order to prepare sequential data for the genetic algorithm , which work with sequences /time series. Creating many small sequences of length sequence_length from my data. For each sequence, I assigned the next target value as the label. This approach is to predict the next value given previous time steps.

4 Genetic Algorithm

Genetic Algorithms are based on the ideas of natural selection and genetics , simulates the natural selection environment where we define chromosomes, selection, crossover and mutation processes in order to get best generation , in which have the better loss value . Each generation consists of a population of individuals. The Following is the approach of GA's :

- Initializing population space where in each generation consist of populations of chromosome
- Selecting fittest parents before crossover and generating new population
- Performing mutation for natural selection on new population
- Moving to new generation with the best chromosome

4.1 Chromosome

Chromosome is an array of candidate solution to the problem.

```
Chromosomes should be consist of sequence_lenght * num_features
```

4.2 Selection

After defining chromosome , in Selection stage, we are deciding fittest parents to be used to produce next generation. In order to do that, we first define *model_builder* function where we modeled our neural network to get the loss score comparing predicted and expected values. After we received our loss values, $fitness\_function$ is defined to calculate fitness value of the individual.

$$f(x) = \frac{1}{1+x} \tag{1}$$

4.2.1 Roulette Selection

Roulette Selection is a probabilistic method used to select individuals based on their fitness values. After we get the fitness values, each individual in the population is assigned a portion of a roulette wheel proportional. The mathematical function we used to find the portion is

$$P(i) = \frac{p(i)}{\sum_{j=1}^{n} p_j} \tag{2}$$

This method allows higher quality individuals to be chosen often.

4.2.2 Tournament Selection

Tournament Selection works by randomly selecting a small group of individuals from the population called a tournament and then choosing the best one among them to be a parent. This process is repeated until the desired number of parents is selected.

4.3 Crossover

Crossover is a genetic operator used to combine the genetic information of two parent individuals to produce new offspring. It mimics biological reproduction by exchanging segments of the parents' chromosomes. Hyperparameter crossover probability is used to apply cross over in a probability.In our Task , we used one point cross over where we selected randomly a point of indices of a one parent length, then we swap the tails of two parents between each other to have 2 new children.

4.4 Mutation

Mutation is a genetic operator that introduces small random changes to an individual's chromosome. It is applied with a low probability (mutation rate) to avoid losing high quality chromosome and is crucial for maintaining genetic diversity within the population. In our task, we randomly select a point within the length of the individual chromosome. Then we change the value at this point to

$$1 - value \tag{3}$$

, effectively reversing its value since our values are normalized.

5 Evaluation

The hyperparameters used in order to evaluate genetic algorithm are the following:

```
hyperparameters={
  'generation_size':[10,20],
  'crossover_p':[0.7,0.9],
  'mutation_p':[0.01,0.05],
  'selection_type':["Roulette","Tournament"],
}
```

5.1 The best loss scores of 10 generations with different Hyperparameter combinations

```
Generation 1 Best Loss: 0.01220269687473774
Generation 2 Best Loss: 0.012171358801424503
Generation 3 Best Loss: 0.012232563458383083
Generation 4 Best Loss: 0.012232286855578423
Generation 5 Best Loss: 0.012185527011752129
Generation 6 Best Loss: 0.012020397000014782
Generation 7 Best Loss: 0.012066668830811977
Generation 8 Best Loss: 0.012063506059348583
Generation 9 Best Loss: 0.012136234901845455
Generation 10 Best Loss: 0.012158486992120743
Final generation complete.
Generation size :10,Crossover prob :0.7, Mutation prob:0.01,Selection type:Roulette
```

```
Generation 1 Best Loss: 0.012185173109173775
Generation 2 Best Loss: 0.01227590162307024
Generation 3 Best Loss: 0.012210311368107796
Generation 4 Best Loss: 0.01220565102994442
Generation 5 Best Loss: 0.012242455035448074
Generation 6 Best Loss: 0.01223670318722725
Generation 7 Best Loss: 0.01222848892211914
Generation 8 Best Loss: 0.012200506404042244
Generation 9 Best Loss: 0.012101378291845322
Generation 10 Best Loss: 0.01216923352330923
Final generation complete.
Generation size :10,Crossover prob :0.7, Mutation prob:0.01,Selection type:Tournament
```

```
Generation 1 Best Loss: 0.012220310047268867
Generation 2 Best Loss: 0.01228410191833973
Generation 3 Best Loss: 0.01218591257929802
Generation 4 Best Loss: 0.012232878245413303
Generation 5 Best Loss: 0.01220141164958477
Generation 6 Best Loss: 0.012150158174335957
Generation 7 Best Loss: 0.011974765919148922
Generation 8 Best Loss: 0.012102626264095306
Generation 9 Best Loss: 0.012190858833491802
Generation 10 Best Loss: 0.012202953919768333
Final generation complete.
Generation size :10,Crossover prob :0.7, Mutation prob:0.05,Selection type:Roulette
```

```
Generation 1 Best Loss: 0.012157704681158066
Generation 2 Best Loss: 0.012045040726661682
Generation 3 Best Loss: 0.012144053354859352
Generation 4 Best Loss: 0.012254471890628338
Generation 5 Best Loss: 0.012152676470577717
Generation 6 Best Loss: 0.01229527872055769
Generation 7 Best Loss: 0.01230567041784525
Generation 8 Best Loss: 0.012110511772334576
Generation 9 Best Loss: 0.012134972028434277
Generation 10 Best Loss: 0.012144605629146099
Final generation complete.
Generation size :10,Crossover prob :0.7, Mutation prob:0.05,Selection type:Tournament
```

```
Generation 1 Best Loss: 0.012167689390480518
Generation 2 Best Loss: 0.012197891250252724
Generation 3 Best Loss: 0.01215196494013071
Generation 4 Best Loss: 0.012020858936011791
Generation 5 Best Loss: 0.012079266831278801
Generation 6 Best Loss: 0.012177247554063797
Generation 7 Best Loss: 0.012187016196548939
Generation 8 Best Loss: 0.012212894856929779
Generation 9 Best Loss: 0.012221571989357471
Generation 10 Best Loss: 0.012216240167617798
Generation 11 Best Loss: 0.012145905755460262
Generation 12 Best Loss: 0.01214676909148693
Generation 13 Best Loss: 0.012070706114172935
Generation 14 Best Loss: 0.012172667309641838
Generation 15 Best Loss: 0.012152871116995811
Generation 16 Best Loss: 0.012159288860857487
Generation 17 Best Loss: 0.012130812741816044
Generation 18 Best Loss: 0.0121145099401474
Generation 19 Best Loss: 0.012094107456505299
Generation 20 Best Loss: 0.01204743143171072
Final generation complete.
Generation size :20,Crossover prob :0.7, Mutation prob:0.01,Selection type:Roulette
```

```
Generation 1 Best Loss: 0.012160871177911758
Generation 2 Best Loss: 0.012224902398884296
Generation 3 Best Loss: 0.012149755842983723
Generation 4 Best Loss: 0.012188605032861233
Generation 5 Best Loss: 0.012140166945755482
Generation 6 Best Loss: 0.012128980830311775
Generation 7 Best Loss: 0.012165924534201622
Generation 8 Best Loss: 0.012223156169056892
Generation 9 Best Loss: 0.012177827768027782
Generation 10 Best Loss: 0.012250789441168308
Generation 11 Best Loss: 0.012074870988726616
Generation 12 Best Loss: 0.012091792188584805
Generation 13 Best Loss: 0.01215595193207264
Generation 14 Best Loss: 0.012191353365778923
Generation 15 Best Loss: 0.012194119393825531
Generation 16 Best Loss: 0.012152470648288727
Generation 17 Best Loss: 0.012183886021375656
Generation 18 Best Loss: 0.01227870024740696
Generation 19 Best Loss: 0.012193001806735992
Generation 20 Best Loss: 0.012138769961893559
Final generation complete.
Generation size :20,Crossover prob :0.7, Mutation prob:0.01,Selection type:Tournament
```

6 References

- Lecture Note: http://gnjatovic.info/machinelearning/
- Dataset: https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction/data
- Code Snippets: http://gnjatovic.info/machinelearning/