|  |  |
| --- | --- |
| 학과 | 컴퓨터 소프트웨어 공학부 |
| 조원 | 201995085 – 정주몽(조장) |
| 201995066 - 이승욱 |



**클라우드 팀플 보고서**

목차

1. 개요
2. 구현한 주요 기능 코드 설명
3. 최종 화면
4. 코드 소스
5. 마무리
6. 개요

**1) 주제:** 달러 환율 예측 AI

**2) 목적**

사용자가 원하는 기간의 달러 환율의 기록을 조회하고, AI모델을 사용해서 AI가 사용자에게 앞으로 3개월 간의 달러 환율의 값을 예측하여 보여주기 위함. 그리고 사용자가 원하는 달러 환율 도달 시 메시지 받기를 받도록 함

**3) 조원 역할**

201995085 – 정주몽(조장):

- AWS Cognito를 이용한 로그인 페이지 만들기

- 환율 데이터를 AI 모델이 학습시키게 하고 학습한 데이터를 바탕으로 앞으로 3달의 환율 값을 예측하게 하는 모델 작성

- html 작성 및 관리

201995066 - 이승욱:

- 환율 데이터 수집과 데이터 전처리를 작성하는 파이썬 코드를 작성

- 사용자가 원하는 달러 환율에 도달할 경우 aws를 이용해 메시지를 보내도록 하는 기능 작성

**4) 기능**

- 로그이 기능

- 사용자가 원하는 기간을 설정 후 설정한 기간의 환율 데이터 값을 보여줌.

- 전처리가 된 데이터를 학습한 AI가 현재시간 기준 3달 뒤의 환율 예측값을 보여줌.

- 사용자가 설정한 환율 값에 도달하면 aws 메시지 기능을 사용하여 메시지를 보냄.

1. 구현한 주요 기능 및 코드

**1. 로그인 기능(작성자: 정주몽)**

**1) 사용자 풀 생성**

텍스트, 스크린샷, 폰트, 디스플레이이(가) 표시된 사진

자동 생성된 설명

**2) 로그인 옵션 설정**

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**3) 사용자 풀 생성 및 사용자 풀 불러오기**

**텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**4) 회원가입 페이지 작성**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명  
5) 로그인 페이지 작성 및 로그인**

**텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 직사각형, 폰트이(가) 표시된 사진

자동 생성된 설명**

**2. 환율 데이터 수집 및 전처리(작성자: 이승욱)**

**1) 환율 데이터 수집 및 데이터 전처리 함수**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**2) 환율 데이터의 기간을 정하기 위해 기간별 환율 데이터 값 수집.**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**3) 데이터 전치가 된 파일은 csv 파일로 저장됨.**

**텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명**

**3. AI 모델 구축과 데이터 학습 및 예측값 도출(작성자: 정주몽)**

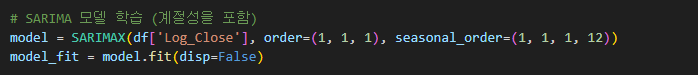
**1) 전처리 된 데이터 불러오는 함수**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**2) 불러온 데이터를 AI 모델에 학습시키는 함수**

****

****

**3) AI가 학습한 데이터를 바탕으로 3개월 후 달러 환율 예측값 추출하는 함수**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**4) 결과 값을 정리 후 csv 파일로 저장하는 함수**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**5) 저장된 csv 파일**

**텍스트, 스크린샷, 번호, 평행이(가) 표시된 사진

자동 생성된 설명**

**4. 웹 페이지 작성(작성자: 이승욱, 정주몽)**

**1) 기간별 달러 환율 그래프 구현 함수(작성자: 이승욱)**

**텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 라인, 그래프, 폰트이(가) 표시된 사진

자동 생성된 설명**

**2) AI 모델이 예측한 달러 환율 값을 그래프로 구현하는 함수(작성자:정주몽)**

**텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명**

1. 최종 구현 화면
2. **로그인 및 회워가입 페이지**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**2) 메인 페이지(10년간 달러 환율, 앞으로 3개월 동안 예측 환율)**

**텍스트, 스크린샷, 그래프, 도표이(가) 표시된 사진

자동 생성된 설명**

**3) 기간별 달러 환율 정보 그래프(1개월간, 6개월간, 1년간, 5년간**

**텍스트, 도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명텍스트, 라인, 폰트, 그래프이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 폰트, 그래프이(가) 표시된 사진

자동 생성된 설명텍스트, 라인, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명**

1. 최종 구현 화면

**1) 소스 코드(Main.html)**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Exchange Rate Visualization</title>

    <!-- Plotly.js를 로드하기 위한 CDN 링크 -->

    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>

</head>

<body>

    <!-- 그래프를 나타낼 div -->

    <div id="exchangeRateGraph"></div>

    <!-- 기간 선택 옵션 태그 -->

    <label for="periodSelect">Select Period:</label>

    <select id="periodSelect" onchange="updateGraph()">

        <option value="1mo">1 Month</option>

        <option value="6mo">6 Months</option>

        <option value="1y">1 Year</option>

        <option value="5y">5 Years</option>

        <option value="10y" selected>10 Years</option> <!-- 기본 선택값 -->

    </select>

    <script>

        // 파이썬에서 생성된 CSV 파일의 그래프를 불러와서 표시하는 함수

        function displayGraph(period) {

            var csvFile = `exchange\_rate\_data\_${period}.csv`; // 기간에 따라 다른 파일명 설정

            Plotly.d3.csv(csvFile, function(err, rows){

                function unpack(rows, key) {

                    return rows.map(function(row) { return row[key]; });

                }

                var dates = unpack(rows, 'Date');

                var rates = unpack(rows, 'Close');

                var data = [{

                    type: 'scatter',

                    mode: 'lines',

                    x: dates,

                    y: rates,

                    name: 'USD/EUR'

                }];

                var layout = {

                    title: `USD/EUR Exchange Rate over the Last ${period}`,

                    xaxis: { title: 'Date' },

                    yaxis: { title: 'Exchange Rate' }

                };

                Plotly.newPlot('exchangeRateGraph', data, layout);

            });

        }

        // 페이지 로드 시 기본 그래프 표시

        displayGraph('10y'); // 기본값 10년 데이터 표시

        // 옵션 선택 변경 시 호출되는 함수

        function updateGraph() {

            var period = document.getElementById('periodSelect').value;

            displayGraph(period); // 선택된 기간에 따라 그래프 업데이트

        }

    </script>

<script>

    // CSV 파일을 로드하여 그래프로 표시하는 함수

    function plotExchangeRate() {

        Plotly.d3.csv('predicted\_exchange\_rate.csv', function(data) {

            var dates = data.map(row => row.Date);

            var forecast = data.map(row => row.Forecast);

            var trace = {

                x: dates,

                y: forecast,

                mode: 'lines',

                line: {color: 'red'},

                name: '예측 환율'

            };

            var layout = {

                title: '환율 예측 (3개월, SARIMA)',

                xaxis: {title: 'Date'},

                yaxis: {title: '환율', range: [0.8, 1.05]},  // 세로 범위 설정

                margin: {l: 50, r: 50, b: 50, t: 50}

            };

            Plotly.newPlot('plot', [trace], layout);

        });

    }

    // 페이지 로드 시 그래프 표시

    plotExchangeRate();

</script>

</body>

</html>

**2) 소스 코드(Data.py)**

from flask import Flask, render\_template, request

import pandas as pd

app = Flask(\_\_name\_\_)

# 환율 데이터를 가져오는 함수

def fetch\_exchange\_rates(base\_currency='USD', target\_currency='EUR', period='10y'):

    import yfinance as yf

    pair = f'{base\_currency}{target\_currency}=X'

    data = yf.download(pair, period=period)

    return data

# 데이터 수집 및 CSV 파일로 저장하는 함수

def save\_exchange\_data(period='10y'):

    exchange\_data = fetch\_exchange\_rates(period=period)

    if not exchange\_data.empty:

        exchange\_data.reset\_index(inplace=True)

        csv\_file = f'exchange\_rate\_data\_{period}.csv'  # 파일명에 기간 추가

        exchange\_data.to\_csv(csv\_file, index=False)

        return csv\_file

    else:

        return None

# 기본 10년 데이터 저장

save\_exchange\_data()

# 다양한 기간에 대해 데이터 저장

periods = ['1mo', '6mo', '1y', '5y', '10y']

for period in periods:

    save\_exchange\_data(period=period)

@app.route('/')

def index():

    return render\_template('index.html', periods=periods)

@app.route('/exchange\_rate', methods=['POST'])

def exchange\_rate():

    period = request.form['period']

    csv\_file = f'exchange\_rate\_data\_{period}.csv'

    df = pd.read\_csv(csv\_file)

    dates = df['Date'].tolist()

    rates = df['Close'].tolist()

    return render\_template('exchange\_rate.html', dates=dates, rates=rates, period=period)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**3) 소스 코드(Ai.py)**

import pandas as pd

import numpy as np

from statsmodels.tsa.statespace.sarimax import SARIMAX

import matplotlib.pyplot as plt

# CSV 파일 로드할 때 날짜 열을 인덱스로 설정

df = pd.read\_csv('exchange\_rate\_data\_10y.csv', parse\_dates=['Date'], index\_col='Date')

# 데이터의 인덱스를 일일 빈도로  리샘플링하여 결측치 채우기

df = df.resample('D').ffill()

# 로그 변환으로 변동성 증가

df['Log\_Close'] = np.log(df['Close'])

# SARIMA 모델 학습 (계절성을 포함)

model = SARIMAX(df['Log\_Close'], order=(1, 1, 1), seasonal\_order=(1, 1, 1, 12))

model\_fit = model.fit(disp=False)

# 예측을 위해 새로운 인덱스 생성 (3개월 예측)

start\_date = df.index[-1] + pd.DateOffset(days=1)

end\_date = start\_date + pd.DateOffset(months=3)

forecast\_index = pd.date\_range(start=start\_date, end=end\_date, freq='D')  # 일일 빈도로 생성

# 예측 수행

forecast\_log = model\_fit.predict(start=len(df), end=len(df) + len(forecast\_index) - 1, typ='levels')

forecast\_log.index = forecast\_index  # 예측 결과에 인덱스 추가

forecast = np.exp(forecast\_log)  # 로그 변환을 되돌리기

# 예측 결과를 데이터프레임으로 변환

forecast\_df = pd.DataFrame({

    'Date': forecast\_index,

    'Forecast': forecast.round(5)  # 소수점 5자리까지 반올림

})

# 예측 결과를 CSV 파일로 저장 (소수점 5자리까지)

forecast\_df.to\_csv('predicted\_exchange\_rate.csv', index=False, float\_format='%.5f')

print("CSV 파일로 예측된 환율을 저장했습니다.")

1. 후기

**1) 201995085 – 정주몽:**

제대로 된 웹페이지를 하나를 구축하고 싶었지만, Ai 모델을 구축하는 과정에서 본인의 능력 부족으로 인해 핵심 기능만 구현된 웹페이지를 만든 것이 아쉽다. 그리고 Ai모델의 정확성을 더 높이기 위해 경제 신문 데이터를 사용해서 예측값을 추출 할려고 했지만 경제 신문 데이터를 사용하면 저작권에 위반이 되어 정확도가 더 높은 AI모델을 만들지 못한 것이 아쉽고, 개인 사정으로 인해 팀프로젝트 완성 기간이 늦어지면서 발표를 못한 것이 아쉽다.

**2) 201995066 – 이승욱:**

AWS SNS를 사용하여 사용자가 지정하는 달러 값에 도달하면 메시지로 알림을 보내는 서비스를 구축할려고 했지만, 달러 값을 지정하고 데이터를 받아 쓰는 과정에서 오류 해결을 하지 못하여 SNS 기능을 구축 못한 것이 아쉽고, 추가하고 싶은 기능은 많았지만 아쉽게 하지 못하여 안타깝고, 할 수 있는 선에서 최선을 다한 것 같다.

**3) GitHub 주소**

https://github.com/WooMong/DollarExchagePredictionAi.git