미국 은행 재무제표 분석 프로젝트

미국 NYSE 상장 은행 98개사 5개년 분기보고서 모델 데이터 전처리 및 머신러닝

5조 - 김윤일, 박영요, 서동옥, 최준혁

대다수의 재무분석이 제조판매업 기업들을 대상으로 하는 가운데, 은행과 같은 금융회사는 제조업과 구별되는 고유한 회계시스템을 가지므로 금융회사만을 대상으로 재무분석을 진행하면 새로 배우는 점이 많을 것으로 판단하였다. 또한 자동화된 수집 및 전처리가 상대적으로 어려운 Raw Data를 효과적으로 처리할 코드를 고안하는 과정 또한 유의미할 것으로 판단해 프로젝트 주제를 선정하였다.

본 연구는 실리콘밸리 은행(SVB)의 파산 사태에서 착안해 재무적으로 부실한 은행들을 재무제표에서부터 가려내는 것이 가능한지 알아보고자, NYSE에 상장된 98개 은행의 손익계산서와 재무상태표의 일부 항목들을 이용해 기초적인 재무분석을 시행한다. 파이썬 함수를 이용하여 웹 크롤링으로 수집한 엑셀 데이터를 추출해 CSV파일로 변환한 뒤, 스케일러를 사용해 머신러닝을 위한 전처리를 진행하였다. 결측치의 경우 선형 보간법을 사용하여 채우고 머신러닝 모델을 이용해 학습을 진행하였다.

I. 프로젝트 일정 및 개발환경

날짜	내용	비고
07.01	브레인스토밍 및 주제 후보선정	
07.02	주제 확정 및 데이터 범위선정	
07.03	SEC 공시 데이터 자동추출 코드 작성 및 데이터 크롤링	
07.04	누락된 데이터 조치, 모델 구상, 미국 재무제표 데이터 특유의 전처리가 난해한 지점들 단순화, 컬럼 선정	**전처리**
07.05	컬럼 선정(계속), 보다 효율적인 재무제표 Raw 데이터 추출 방안 고민	**전처리**
07.06	7월 3일의 자동추출 코드를 사용해 재무제표 엑셀 파일 다운로드, 추출 가능한 항목들을 이용한 효과적 재무분석을 위해 전략 보완	**전처리**
07.07	자동 다운로드 과정에서 누락된 파일 보완, Raw 데이터에서 CSV파일을 추출하는 코드 작성	**전처리**
07.08	크롤링 코드와 CSV추출 코드 보완 및 샘플 데이터셋에 시험적용	**전처리**
07.09	재무제표 항목 수작업 매핑 및 CSV추출 코드 보완	**전처리**
07.10	재무제표 항목 수작업 매핑 및 CSV추출 코드 보완	**전처리**
07.11	자동화를 통한 98개 은행 재무제표 전처리 완료 및 모델링 분석 테스트	
07.12	전처리된 데이터셋에 오류 발견, 데이터셋 변경 및 모델링 분석 테스트와 시각화	

사용 언어	분석 모델	사용 프로그램
➢ Python 3 (100%)	➢ Random Forest ➢ XG Boost ➢ Linear Regression	➢ VSCode ➢ Jupyter Notebook ➢ Google Colaboratory ➢ MS Excel ➢ Chat GPT ➢ Claude

II. Dataset

II-1. Web Crawling

미국의 회계공시는 U.S. SEC(U. S. Securities and Exchange Commission, 미 증권거래위원회)의 EDGAR 전자공시 시스템에 업로드된다. EDGAR에서는 각 기업의 회계공시이력을 연도별, 유형별로 분류해 csv와 api 형태로 제공하지만, 본 연구에서 목표로 하는 Raw Data는 공시이력이 아닌 공시 본문이므로 Excel 형태의 보고서 파일을 추출하기 위해 동적 크롤링 기법을 사용하였다. 웹사이트의 엑셀 파일 다운로드 링크에 접근해 데이터를 추출한다.

Selenium을 사용한 EDGAR의 url 스크래핑 코드 (하이퍼링크)

Selenium을 사용한 엑셀 파일의 특정 시트만 추출하는 코드 (하이퍼링크)

4쿼터의 정적 크롤링 코드 (하이퍼링크)

II-2. Raw Data

미국 재무제표 데이터는 기업별로 계정의 분류방식과 용어 표기가 다르며, 같은 기업의 경우에도 회계 정책의 변동에 따라 연도별 보고 형식이 상이한 경우가 있어 전처리가 매우 난해하다. 따라서 일반화가 상대적으로 쉬우면서도 최대한 유의미한 재무비율을 산출할 수 있는 계정과목들을 선정하였다.

재무비율을 통한 재무분석은 일반적으로 손익계산서, 재무상태표, 현금흐름표 항목들로 진행이 가능하다. 이 중 손익계산서와 재무상태표 항목들의 일반화가 상대적으로 쉬우므로, 해당 자료들의 항목들 중 비슷한 표기와 형식으로 나타나는 항목들을 선별하였다. NYSE 상장되고 SEC EDGAR에 등록된 98개 은행의 5년 분기보고서(10-Q) 데이터 중 손익계산서와 재무상태표 내의 아래 항목을 수집하였다.

<표 1: 추출 항목>

손익계산서 항목	재무상태표 항목
➢ interest income (이자수익) ➢ net interest income (순 이자수익) ➢ non-interest income(비이자수익) ➢ income before tax (세전수익) ➢ tax expense (법인세비용) ➢ net income (순이익)	➢ cash [and other cash equivalents] (현금 [및 현금성자산]) ➢ goodwill (영업권) ➢ loans (대출)* ➢ net loans (순대출)* ➢ total assets (총자산) ➢ total liabilities (총부채) ➢ total [shareholders'/stockholders'] equity (총자본) ➢ retained earnings (이익잉여금. 유보이익이라고도 함)

*: 원 표기상의 전처리 한계로 인한 일부 누락값 존재

II-2. 데이터 추출

<표 1>의 항목들도 기업마다 표기에 차이가 있는 점을 고려, 정규표현식을 사용한 함수를 이용해 기업별로 <표 1>의 항목들에 대응하는 표기들을 매핑하였다.
(https://colab.research.google.com/drive/1TaKHrrk8vX9Y5zsukiHWmcn7oE6460Lg)

```python
# 2. 특정 파일과 시트에서 금융 데이터를 추출하는 함수
def extract_financial_data(file_path, sheet_names, data_items, date_columns):
    results = []
    try:
        xls = pd.ExcelFile(file_path)
        available_sheets = xls.sheet_names

        for sheet_name in sheet_names:
            if sheet_name in available_sheets:
                df = pd.read_excel(file_path, sheet_name=sheet_name, header=[0, 1])
                df.columns = [f'{col[0]} {col[1]}' if 'Unnamed' not in col[1] else col[0] for col in df.columns]

                # 시트 이름에 특정 단어가 포함된 경우 확인
                unit_in_thousands = any('in Thousands' in col for col in df.columns)
                unit_in_millions = any('in Millions' in col for col in df.columns)

                # 각 항목에 대해 패턴을 기반으로 데이터를 추출
                items = data_items.get(sheet_name, {})

                for item, pattern in items.items():
                    row = find_row_by_pattern(df, pattern)
                    if row is not None:
                        for date in date_columns:
                            matching_col = [col for col in df.columns if re.search(date, col, re.IGNORECASE)]
                            if matching_col:
                                col_name = matching_col[0]
                                value = row.get(col_name, None)
                                if value is not None:
                                    try:
                                        # 괄호 처리를 통해 음수 값 변환 및 수치 변환
                                        if isinstance(value, str) and '(' in value and ')' in value:
                                            value = -float(value.replace('(', '').replace(')', '').replace(',', ''))
                                        elif isinstance(value, str):
                                            value = float(value.replace(',', ''))
                                        # 단위 조정
                                        if unit_in_thousands:
                                            value = round(value / 1000, 1)  # Thousands to Millions 변환
                                        elif unit_in_millions:
                                            value = round(value, 1)  # Millions 단위 그대로 유지
                                        else:
                                            value = round(value / 1000000, 1)  # Dollars to Millions 변환
                                        results.append({
                                            'Date': date,
                                            'Item': item,
                                            'Value': value
                                        })
```

II-3. 데이터 전처리

라벨 인코더, 원핫 인코더와 min-max 스케일러를 사용해 엑셀 파일로부터 추출한 Raw data를 스케일링 하였다.

Name	Year	Quarter	Cash and Due from Banks	Goodwill	Income Before Tax	Interest Income	Loans	Net Income	Net Interest Income	Net Loans	Non-Interest Income	Retained Earnings	Tax Expense	Total Assets
BANF	2018	Mar. 31	181.9	79.8	36.9	70.9	4984.5	29.6	63.0	4932.9	30.1	661.3	7.3	7615.6
BANF	2018	Jun. 30	188.5	79.7	39.8	75.1	5007.5	30.6	64.9	4955.3	30.4	684.4	9.2	7623.0
BANF	2018	Sep. 30	185.0	79.7	41.9	77.4	4947.5	32.9	65.7	4895.7	32.8	707.5	9.0	7602.4
BANF	2018	Dec. 31	228.4	79.7	41.2	79.8	4976.0	32.7	66.9	4924.6	31.9	722.6	8.4	7574.3
BANF	2019	Mar. 31	186.0	79.7	41.0	80.9	5042.5	31.8	66.9	4989.6	32.0	744.7	9.2	7709.0
BANF	2019	Jun. 30	185.4	79.7	43.8	83.1	5094.4	34.2	68.8	5039.3	34.1	769.1	9.7	7642.0
BANF	2019	Sep. 30	225.5	147.0	43.0	86.4	5606.8	33.4	72.3	5550.9	35.6	792.0	9.6	8388.8
BANF	2019	Dec. 31	222.0	148.6	41.8	86.3	5662.1	69.6	73.9	5607.9	35.5	815.5	6.2	8565.8
BANF	2020	Mar. 31	192.0	149.9	28.2	84.0	5989.9	22.6	74.1	5919.8	35.1	826.9	5.6	8669.1
BANF	2020	Jun. 30	205.2	149.9	25.3	81.5	6675.0	20.7	77.2	6585.5	32.1	837.2	4.6	9612.5
BANF	2020	Sep. 30	226.1	149.9	25.6	79.2	6612.1	20.9	75.9	6506.0	34.6	846.9	4.7	9618.9
BANF	2020	Dec. 31	280.5	149.9	44.4	82.4	6394.5	35.4	79.5	6303.1	35.4	871.2	9.0	9212.4
BANF	2021	Mar. 31	274.1	149.9	52.2	80.0	6358.4	42.5	77.2	6267.5	39.9	898.0	9.7	10549.3
BANF	2021	Jun. 30	268.3	149.9	62.9	84.9	6191.2	48.2	82.4	6107.3	44.6	935.1	14.7	11015.3
BANF	2021	Sep. 30	274.1	149.9	48.3	83.2	6016.9	38.8	80.2	5930.5	39.8	950.6	9.5	11302.8

Cash and Due from Banks	Goodwill	Income Before Tax	Interest Income	Loans	Net Income	Net Interest Income	Net Loans	Non-Interest Income
-0.351369612674923	-0.448698908169867	-0.200131744687107	-0.429703964829397	-0.40627377076912	-0.167801137468657	-0.458667137482513	-0.454440538220538	-0.168980414134821
-0.341958690494684	-0.448793832178786	-0.194424673220224	-0.421454384095869	-0.40552481367397	-0.165786246378082	-0.453354428502287	-0.453691484107609	-0.168428076240917
-0.346949331044811	-0.448793832178786	-0.190291966295929	-0.416936756551319	-0.407478614791752	-0.161151996869759	-0.45111749840535	-0.455684503086652	-0.164009373089681
-0.285065388223239	-0.448793832178786	-0.191669535270693	-0.412222710417874	-0.406550559260806	-0.161554975087874	-0.447762103259945	-0.454718089521311	-0.165666386771395
-0.345523433744775	-0.448793832178786	-0.192063126406341	-0.410062105940046	-0.404385096355265	-0.163368377069392	-0.447762103259945	-0.452544494997187	-0.165482274140093
-0.346378972124796	-0.448793832178786	-0.18655285050728	-0.405740896984388	-0.402695058388384	-0.158532638452011	-0.44244939427972	-0.450882531184127	-0.161615908882762
-0.289200490393344	-0.384909974176242	-0.188127215049869	-0.399259083550902	-0.386009596842533	-0.160144551324471	-0.43266282510562	-0.433774670283487	-0.158854219413239
-0.294191130943471	-0.383391190033537	-0.190488761863752	-0.399455502139796	-0.384208843478978	-0.0872054938456449	-0.428188964911746	-0.431868595085409	-0.159038332044541
-0.336968049944557	-0.382157177917589	-0.217252959087759	-0.403973129684347	-0.373534576705501	-0.181905375102685	-0.427629732387512	-0.421438685361192	-0.159774782569747
-0.318146205584079	-0.382157177917589	-0.222960030554642	-0.408883594406684	-0.351225424275636	-0.185733668174778	-0.418961628261881	-0.399177733442592	-0.165298161508792
-0.288344952013322	-0.382157177917589	-0.222369643851172	-0.413401221951235	-0.35327365911411	-0.185330689956663	-0.422596639669404	-0.401836206745173	-0.160695345726254
-0.210776138891353	-0.382157177917589	-0.185372077100339	-0.407115827106643	-0.360359444501263	-0.156114769143321	-0.412530454233187	-0.408621165652014	-0.159222444675842
-0.219901881611584	-0.382157177917589	-0.1700220228101	-0.411829873240087	-0.361534981507128	-0.141809042400236	-0.418961628261881	-0.409811626652919	-0.150937376267275
-0.228172085951794	-0.382157177917589	-0.148964897052977	-0.402205362384305	-0.366979573955345	-0.130324163183957	-0.404421582631791	-0.41516870115699	-0.142284082596105
-0.219901881611584	-0.382157177917589	-0.17769704995522	-0.405544478395495	-0.372655366202499	-0.149264139435365	-0.410573140398368	-0.421080878262605	-0.151121488898577
-0.284495029303225	-0.382157177917589	-0.184191303693398	-0.413990477717916	-0.367689455028139	-0.150674563198767	-0.422596639669404	-0.415897691320465	-0.140258843651789
-0.218761163771555	-0.356812467536193	-0.186749646075104	-0.41477615207349	-0.357109621975354	-0.155107323598033	-0.423715104717872	-0.405143414413417	-0.14412520890912
-0.198656011841045	-0.350167786911857	-0.164118155775392	-0.389241735517333	-0.353234583091755	-0.13737628200097	-0.39183885083652	-0.401157376455332	-0.145966335222135
-0.28706164444329	-0.351591647045643	-0.138337936390504	-0.345047553016292	-0.346249744095687	-0.115816947331814	-0.352692574140123	-0.394081490281416	-0.133630788924935
-0.241432930842131	-0.351591647045643	-0.134598820601856	-0.302228300637506	-0.342478907938369	-0.112391632477836	-0.326129029238996	-0.390306123792592	-0.135656027869251
-0.307309386103804	-0.351591647045643	-0.126530202321089	-0.283372116103729	-0.336780321344841	-0.111585676041606	-0.329484424384402	-0.384521018366848	-0.136392478394458
-0.295474438513504	-0.351591647045643	-0.134992411737503	-0.272765512303479	-0.330915661656301	-0.116622903768045	-0.338711761034267	-0.378568713362325	-0.136024253131855

III. Machine Learning Model

III-1. 실험 과정

머신러닝에는 3가지 모델을 사용하였다. 머신러닝 시행 기록 (하이퍼링크)

<표 2: <표 1>의 재무제표 항목들과 주가 간 상관관계 예측 모델>

타겟: 주가	Linear Regression	Random Forest	XG Boost
MSE	0.3861	0.1153	0.1275
RMSE	0.6214	0.3395	0.3571
R2 Score	0.5199	0.8566	0.8414
F1 Score	0.4615	0.4828	0.7619

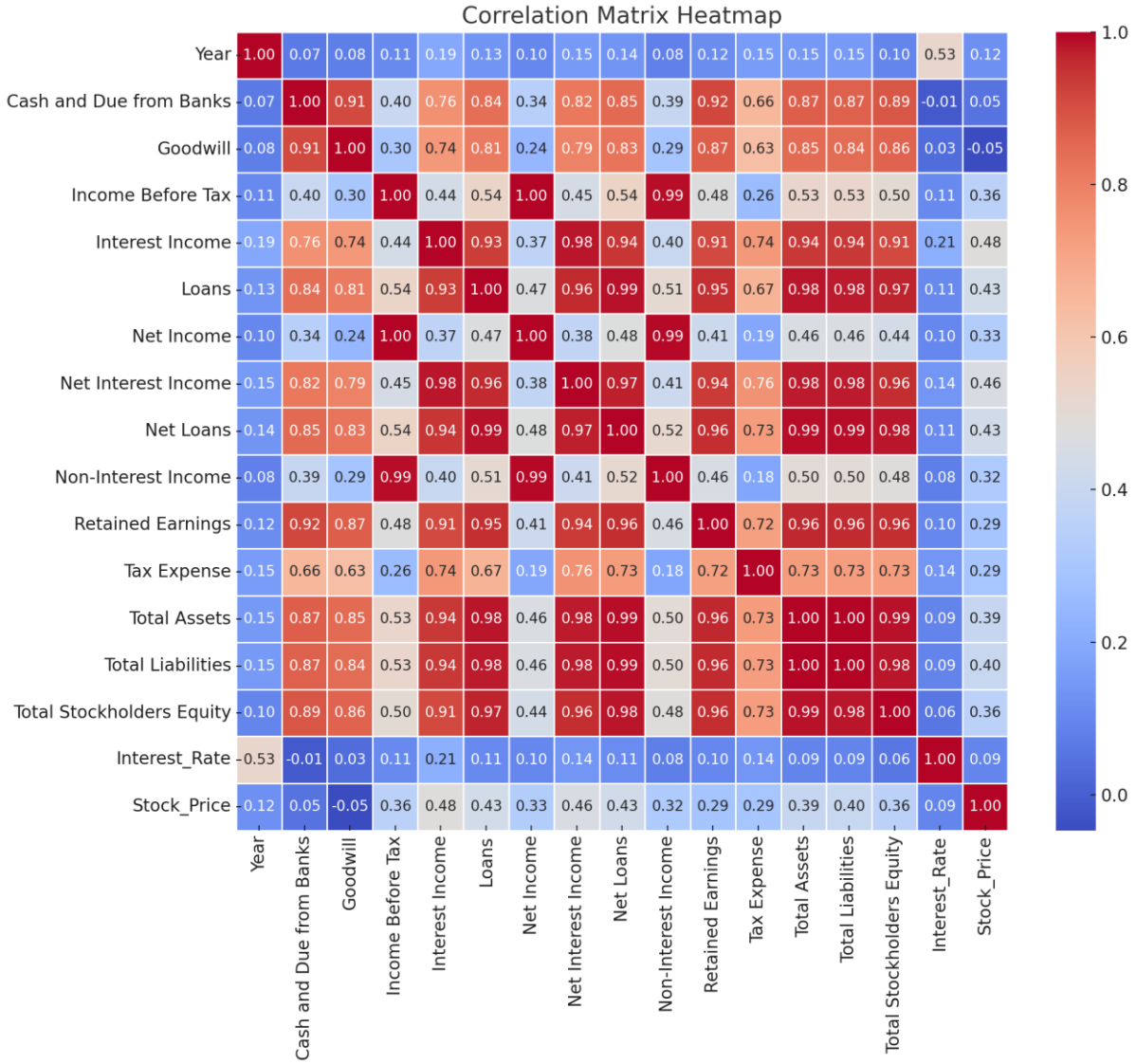
<표 3: <표 1>의 재무제표 항목들과 금리 간 상관관계 예측 모델>

타겟: 금리	Linear Regression	Random Forest	XG Boost
MSE	2.1956	0.3185	0.1904
RMSE	1.4818	0.5644	0.4363
R2 Score	0.3659	0.9080	0.9450
F1 Score	0.8333	0.9667*	0.9667*

*Tree 구조에선 F1 score 값이 원래 나오지 않지만 나오면서 생긴 오류로 추정

III-2. 시각화

전체적인 상관관계 히트맵을 먼저 출력한 뒤 선 그래프로 대조, Impact factor를 확인하였다.



IV. 프로젝트의 의의 및 시사점

전처리가 지나치게 난해한 데이터셋을 선정한 점이 아쉬웠다. 매핑만 완료되면 자동화가 가능했으나, 표기와 형식이 상이한 동시에 방대한 Raw Data였으므로 수작업으로 매핑하였고, 이는 매우 비효율적이었다.

팀원 간 도메인 지식의 교집합이 매우 적고, 사전조사가 충분치 않았던 것도 이유라고 생각한다. 그러나 모델의 결과물에서 유의미한 상관관계를 도출할 수 있었기 때문에, 조금 더 시간을 가지고 제대로 작업할 수 있다면 유용한 모델을 만들 수 있었을 것으로 기대된다.