

숫자로 읽는 기업

성장성과 수익성 분석부터 예측 모델링까지

강의 전체 목표 및 로드맵

강의 목표

✅ 데이터 기반 투자 사고방식 훈련

- 감(感)이 아닌, 숫자와 데이터로 기업을 분석
- 재무제표 기반의 기업 경쟁력 평가
- 기업의 성장성과 수익성을 데이터로 해석하는 눈 키우기

✅ 데이터 분석부터 대시보드까지 완성

- 실제 5,000개 기업 비식별 재무 데이터셋 활용
- 데이터 수집 → 저장 → 분석 → 시각화 → 대시보드 완성
- Colab + DuckDB + Streamlit 으로 현업형 분석 파이프라인 구축

✅ 예측 모델링까지 확장

- 과거 데이터를 기반으로 미래 매출 및 수익성 예측
- 분류/회귀 모델을 활용하여 투자 의사결정 보조

학습 기대 효과

🎯 기업 재무 분석 역량 확보

- 매출 증가율, ROE, 부채비율 등 주요 지표의 계산과 해석 능력

🎯 데이터 기반 투자 판단

- 재무 지표로 기업의 현재 위치와 성장 가능성 진단

🎯 실무형 분석 환경 익히기

- Colab 과 DuckDB 를 활용하여 대용량 데이터 처리
- Streamlit 으로 실시간 대시보드 구현 및 공유

🎯 예측 모델을 통한 투자 전략 보완

- 머신러닝 모델로 성장성 예측 및 리스크 관리

왜 예측이 필요한가?

1 데이터 분석 → 인사이트 도출

- 데이터 분석(Descriptive Analytics)은 과거와 현재의 데이터를 정리하고, 의미를 해석하는 과정입니다.
- 기업의 재무 데이터 분석을 통해 매출 추이, 이익률 변화, 비용 구조 등을 파악할 수 있습니다.
- 이 과정은 주로 다음의 질문에 답합니다.
 - ✓ "기업의 매출과 수익성은 어떤 패턴을 보였는가?"
 - ✓ "주요 비용 항목이 이익률에 어떤 영향을 미쳤는가?"
 - ✓ "경쟁사 대비 수익 구조에 차이가 존재하는가?"

💡 **요약:** 예측 분석은 불확실한 미래 상황을 정량화하여, 보다 합리적인 의사결정을 가능하게 합니다.

"데이터 분석은 일종의 건강검진입니다. 기업의 재무 상태를 진단해서, 어디가 건강하고 어디가 약한지 먼저 파악하는 거죠."

왜 예측이 필요한가?

2 예측 분석 → 미래 의사결정 지원




- 예측 분석(Predictive Analytics)은 데이터 분석의 연장선으로, 미래의 가능성 있는 시나리오를 수치화합니다.
- 통계적 방법론 및 기계학습 모델을 활용하여, 주요 재무 지표의 향후 변화를 추정할 수 있습니다.
- 이는 경영 전략 및 투자 의사결정에 중요한 기초 자료가 됩니다.
- 주요 활용 질문:
 - ✓ "향후 매출 성장률은 어느 수준이 될 것인가?"
 - ✓ "수익성 지표(예: 영업이익률, ROE)는 개선될 가능성이 있는가?"
 - ✓ "재무 건전성은 유지될 수 있는가? (부채비율 예측 등)"

💡 **요약:** 예측 분석은 불확실한 미래 상황을 정량화하여, 보다 합리적인 의사결정을 가능하게 합니다.

"여기서 중요한 것은 '가능성'입니다. 예측 모델은 절대적인 답이 아니라, '이런 방향성이 유력하다'는 신호를 제공하는 도구입니다."

왜 예측이 필요한가?

3 기업 분석에서의 예측 분석 실용 사례

활용 영역	설명
 매출 성장 예측	과거 매출 데이터와 시장 변수를 이용하여 향후 매출 증가율을 예측합니다. 기업의 성장성을 사전에 판단할 수 있습니다.
 수익성 개선 여부 판단	이익률, 원가 구조, 판관비 변화를 바탕으로 기업의 이익 창출 능력의 향후 방향성을 예측합니다.
 투자 전략 및 포트폴리오 설계	예측 모델 결과를 바탕으로 성장 가능성이 높은 기업을 선별하고, 분산 투자를 통한 리스크 관리 전략을 수립합니다.

💡 **요약:** 예측 분석은 단일 기업 분석뿐 아니라, 투자 전략 및 포트폴리오 최적화에도 유용하게 적용됩니다.

"결국 우리가 예측하는 이유는 투자나 경영 전략을 더 똑똑하게 짜기 위함입니다. 감에 의존하는 시대는 지났습니다!"

1 예측 모델이란?

- 예측 모델(Predictive Model)은 과거 데이터를 활용하여 미래의 결과값 또는 아직 관측되지 않은 데이터를 추정하는 통계적/수학적 도구입니다.
- 입력 변수(Input, Feature)를 바탕으로 목표 변수(Target, Output)를 추정합니다.
- 주요 활용
 - 재무 지표 예측 (예: 매출 성장률, 영업이익률)
 - 수요 예측
 - 위험 분석 (Risk Modeling)

💡 **요약:** 예측 모델은 '과거로 미래를 학습하는' 도구입니다.

"예측 모델은 과거 데이터를 수학적으로 학습하여, 미래의 불확실성을 수치로 표현하는 체계입니다. 불확실성을 완전히 제거할 수는 없지만, 합리적으로 축소할 수 있습니다."

2 지도학습 (Supervised Learning) 개념

- 지도학습(Supervised Learning)은 정답(label) 이 주어진 데이터를 기반으로 학습하는 방식입니다.
- 입력 데이터 (X) 와 출력 값 (Y) 의 쌍을 학습하여, 새로운 입력 데이터에 대해 Y 값을 예측합니다.
- 지도학습의 목적:
 - 숨겨진 패턴을 찾아 미래 데이터를 예측
 - 복잡한 데이터 간의 관계를 수학적 함수로 모델링

💡 **요약:** 지도학습은 '답이 있는 데이터' 를 통해 학습하고 미래를 예측합니다.

"지도학습을 이해하는 가장 쉬운 방법은 선생님이 학생에게 문제를 풀게 하고, 정답지를 통해 학습하는 과정이라고 생각하시면 됩니다."

3 지도학습(Supervised Learning)의 두 가지 축

- 지도학습(Supervised Learning) 은 "정답(label)이 있는 데이터"를 기반으로 미래를 예측하는 방법입니다.
- 지도학습 문제는 출력값(Target)의 형태 에 따라 두 가지로 나뉩니다.

구분	설명
분류 (Classification)	출력값이 범주형 데이터 — ex) ‘흑자/적자’, ‘위험/안전’ 등
회귀 (Regression)	출력값이 연속형 수치 데이터 — ex) ‘매출 증가율: 15.3%’, ‘영업이익률: -4.5%’ 등

4 분류 (Classification)

- 문제 유형: 범주형 결과를 예측
- 예시:
 - "이 기업은 흑자를 낼 것인가, 적자를 낼 것인가?"
 - "기업의 재무 건전성은 위험군인가, 안전군인가?"
 - "등급 평가: AAA / AA / A / BBB ..."
- 주요 모델:
 - 로지스틱 회귀 (Logistic Regression)
 - 의사결정나무 (Decision Tree)
 - 랜덤포레스트 (Random Forest)
 - XGBoost Classifier

"분류 모델은 기업을 그룹으로 나누고자 할 때 유용합니다. 예를 들어 '부실 가능성이 높은 기업' 을 선별하는 작업이죠."

5 회귀 (Regression)

- 문제 유형: 연속형 수치 결과를 예측
- 예시:
 - "내년도 매출 증가율은 몇 %일까?"
 - "영업이익률은 몇 %로 전망되는가?"
 - "부채비율은 얼마나 될 것인가?"
- 주요 모델:
 - 선형 회귀 (Linear Regression)
 - 랜덤포레스트 회귀 (RandomForestRegressor)
 - XGBoost Regressor
 - 다항 회귀 (Polynomial Regression)

"경영에서 수치 예측은 매우 중요합니다. 매출이 10% 오를지 50% 오를지에 따라 투자 전략이 완전히 달라지니까요."

예측 모델링 개념 이해

예측 목적에 따라 달라지는 모델링 접근

6 분류와 회귀의 주요 차이점

구분	분류 (Classification)	회귀 (Regression)
출력 값 (Target)	범주형 (Class, Category)	연속형 수치 (Continuous Value)
목적	그룹(카테고리) 구분	값(수치) 예측
예시 문제	기업 위험 등급 예측	매출 성장률 예측
주요 평가 지표	정확도 (Accuracy), F1 Score	RMSE, MAE, R ² (결정계수)
실습 적용	기업 부도 예측	매출, 이익률 예측 (이번 실습)
학습 난이도	상대적으로 낮음 (소규모 데이터 시)	데이터 품질과 피처 엔지니어링 중요

"분류는 'Yes or No', 회귀는 '얼마나?' 를 묻는 것입니다. 이번 기업 예측 실습에서는 수치를 예측하는 회귀 모델이 적합합니다."

예측 모델링 개념 이해

예측 목적에 따라 달라지는 모델링 접근

7 기업 분석 실전 적용 관점

기업 분석 시나리오	모델 종류	설명
"기업이 흑자를 낼 확률은?"	분류	흑자/적자 분류 (이진 분류)
"내년도 매출 증가율은?"	회귀	% 수치 예측
"재무 건전성 등급 분류"	분류	신용등급 분류 등
"부채비율 추정"	회귀	연속 수치 예측

"기업 분석에서는 두 모델이 모두 필요합니다. 하지만 오늘 우리가 집중할 것은 바로 수치 기반의 예측, 회귀입니다."

예측 모델링의 단계

데이터에서 인사이트로: 예측 모델링 5단계 프로세스

전체 프로세스 개요

- 데이터 준비 (Data Preparation)
- 피처 엔지니어링 (Feature Engineering)
- 모델 학습 (Model Training)
- 모델 평가 (Model Evaluation)
- 결과 해석 및 활용 (Interpretation & Application)

"모델링은 복잡하지만 체계적인 과정입니다. 각 단계의 목적과 방법을 이해하면, 누구나 데이터 기반 의사결정을 할 수 있습니다."

(1) 데이터 준비

✓ 핵심 포인트

- 신뢰할 수 있는 데이터 소스 확보
 - 공공 데이터, 재무제표, 시장 데이터 등
- 결측치(Missing Value) 처리
 - 삭제 또는 대체 (평균값, 전년도 값 등)
- 이상치(Outlier) 검토
 - 비정상적으로 높은/낮은 값 식별 및 처리
- 시계열(Time Series) 정렬
 - 연도별/분기별 순서 정렬로 정확한 패턴 인식

"모델링의 절반은 데이터 준비입니다. 데이터가 부실하면 모델이 아무리 좋아도 무용지물이 됩니다."

(2) 피처 엔지니어링 (Feature Engineering)

✓ 핵심 포인트

- 입력 변수 가공 및 생성
 - 매출 증가율, 이익률, 부채비율 등
- Lag feature (전년도 변수) 추가
 - 전년도 수치로 패턴 예측
- 업종 평균 대비 차이
 - 기업 개별 성과와 업종 트렌드 비교
- 성장률, 비율 등 파생 변수 생성
 - 기존 데이터에서 새로운 인사이트 도출
- 다중공선성(Multicollinearity) 검토
 - 변수 간 강한 상관관계는 모델 성능 저하 요인

(3) 모델 학습 (Model Training)

✓ 핵심 포인트

- 모델 선택
 - 이번 실습에서는 RandomForestRegressor
- 비선형 데이터에 강하고, 변수 중요도 해석 가능
 - 학습 데이터 / 검증 데이터 분리
- Train-Test Split
 - 일반적으로 80:20 또는 70:30
- 모델 파라미터 튜닝
 - 예: 결정 트리 수, 최대 깊이 등

"모델 선택은 레시피 선택과 비슷합니다. 재료가 같아도 어떤 요리법을 쓰느냐에 따라 결과가 달라집니다."

(4) 모델 평가 (Model Evaluation)

✓ 핵심 포인트

- 예측 성능 검증
 - RMSE (Root Mean Squared Error)
 - 예측값과 실제값 간의 평균적인 오차
 - MAE (Mean Absolute Error)
 - 예측값과 실제값 차이의 절대값 평균
- 과적합(Overfitting) 여부 확인
 - 학습 데이터와 검증 데이터 성능 차이 검토

"모델 평가 지표를 모르면 모델의 신뢰성을 평가할 수 없습니다. 반드시 검증합시다."

(5) 결과 해석 및 활용 (Interpretation & Application)

✓ 핵심 포인트

- 예측 결과 해석
 - 예측값이 의미하는 바를 해석
 - 매출 증가율 15% → 고성장 기업
- 주요 변수(Feature Importance) 분석
 - 모델이 어떤 변수를 중요하게 평가했는가?
- 비즈니스 전략 수립
 - 예측 결과를 투자 전략, 경영 계획에 반영

"예측 결과는 숫자 그 자체가 아닙니다. '그래서 어떤 전략을 세울 것인가?' 가 진짜 질문입니다."

예측 실습 데이터셋 준비

데이터셋 준비하기

✓ 데이터셋 개요

- 총 10,000개 이상 기업
- 다양한 업종
 - 응용 소프트웨어 개발 및 공급업
 - 전자상거래 소매업
 - 시스템 소프트웨어 개발 및 공급업 등
- 분석 기간: 최근 5년 (2019 ~ 2023)
- 기업별 데이터
 - 기업명, 사업자번호, 법인번호
 - 기준연도, 기업 상태 (정상/휴업/폐업 등)

✓ 포함된 주요 재무 항목

손익계산서 (Income Statement)

- 매출액
- 매출원가
- 매출총이익
- 판매비
- 영업이익
- 영업외수익/비용
- 당기순이익

재무상태표 (Balance Sheet)

- 유동자산 / 비유동자산 / 총자산
- 유동부채 / 비유동부채 / 총부채
- 자본금 / 자본잉여금 / 이익잉여금 / 총자본

자본변동표 (Statement of Changes in Equity)

- 자본조정
- 기타포괄손익누계액

기타

- 근로자수 / 입사자수 / 퇴사자수
- 업종명

기업명,사업자번호,법인번호,기준연도,상태,손익계산서_매출액,손익계산서_매출원가,손익계산서_매출총이익,손익계산서_판매비,손익계산서_영업이익,손익계산서_영업외수익,손익계산서_영업외비용,손익계산서_법인세차감전순이익,손익계산서_계속사업손익법인세비용,손익계산서_당기순이익,재무상태표_유동자산,재무상태표_비유동자산,재무상태표_총자산,재무상태표_유동부채,재무상태표_비유동부채,재무상태표_총부채,재무상태표_자본금,재무상태표_자본잉여금,재무상태표_이익잉여금,재무상태표_총자본,자본변동표_자본조정,자본변동표_기타포괄손익누계액,근로자수,입사자수,퇴사자수,업종명

"우리가 사용할 데이터는 실전 데이터입니다. 수치 하나하나가 실제 기업의 움직임을 나타냅니다."

🎯 타겟 변수(Target Variable) 선정

- 이번 실습의 목표: 기업의 미래 성장성과 수익성 예측
- 타겟 변수:
 - 📈 매출 증가율 (성장성 지표)
 - 💰 영업이익률 (수익성 지표)
- 선정 이유:
 - 매출 증가율은 기업의 성장성 을 나타내는 대표적인 지표
 - 영업이익률은 기업의 수익성 을 나타내는 대표적인 지표
 - 기업의 전반적인 재무 건전성을 파악하는 데 핵심적

"우리는 실제 투자나 경영 판단에서 가장 많이 사용하는 두 지표를 예측합니다. 이 지표들이 바로 기업의 경쟁력을 말해줍니다."

피처(Feature) 구성

- **기본 재무 지표**
 - 매출액, 영업이익, 자산총계, 부채총계, 자본총계 등
- **파생 변수 (Feature Engineering)**
 - 전년도 대비 변화율 (Lag Feature)
 - 업종 평균 대비 차이
 - 재무 비율 지표
 - 유동비율 = 유동자산 / 유동부채
 - 부채비율 = 부채총계 / 자본총계
 - 자기자본이익률(ROE) = 당기순이익 / 자본총계
- **인력 지표**
 - 근로자수, 입사자수, 퇴사자수

"좋은 피처는 모델의 눈을 밝힙니다. 우리가 어떻게 피처를 구성하는지가 예측력의 열쇠입니다."

과정 요약

1 Train-Test Split

- 학습 데이터와 검증 데이터를 구분 (예: 80:20 비율)

2 모델 학습 (Training)

- RandomForestRegressor 모델을 사용
- 피처와 타겟 변수 정의
- 모델 훈련

3 예측 및 검증 (Prediction & Evaluation)

- 검증 데이터셋으로 예측 수행
- RMSE, MAE 등 평가 지표로 성능 검토
- 과적합 여부 확인

"학습 데이터로 너무 좋은 성능이 나오더라도 안심할 수 없습니다. 반드시 테스트 데이터로 검증이 필요합니다."

모델 학습 및 검증

기업 성장성·수익성 예측 모델 학습 및 검증

🧩 라이브러리 설치 & 데이터 로딩

```
!pip install -U scikit-learn
```

```
# ✅ 1. 라이브러리
import pandas as pd
import numpy as np
import duckdb
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import font_manager as fm
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import StandardScaler, LabelEncoder

sns.set(style="whitegrid")

# 폰트 설정
font_path = '/usr/share/fonts/truetype/nanum/NanumGothic.ttf'
font_prop = fm.FontProperties(fname=font_path)
plt.rcParams['font.family'] = font_prop.get_name()
plt.rcParams['axes.unicode_minus'] = False

print("🚀 글로벌 모델 학습 및 예측 프로세스 시작!")

# ✅ 2. 데이터 로딩
print("📦 DuckDB 에서 데이터 로딩 중...")
con = duckdb.connect(database='company_data.duckdb', read_only=False)
df = con.execute("SELECT * FROM company_data").df()
con.close()
print(f"✅ 데이터 로딩 완료! 총 {len(df)} 행")
```

모델 학습 및 검증

기업 성장성·수익성 예측 모델 학습 및 검증

🧩 피처 엔지니어링

- 전년 대비 증감률 & 업종 평균 대비 차이 생성
- 로그 변환으로 이상치 영향 완화

```
# ✅ 3. 피처 엔지니어링
df = df.sort_values(by=['사업자번호', '기준연도'])
group = df.groupby('사업자번호')

# 전년도, 2년 전 피처
df['전년_매출액'] = group['손익계산서_매출액'].shift(1)
df['2년전_매출액'] = group['손익계산서_매출액'].shift(2)
df['전년_자산총계'] = group['재무상태표_자산총계'].shift(1)
df['2년전_자산총계'] = group['재무상태표_자산총계'].shift(2)
df['전년_부채총계'] = group['재무상태표_부채총계'].shift(1)
df['2년전_부채총계'] = group['재무상태표_부채총계'].shift(2)
df['전년_근로자수'] = group['근로자수'].shift(1)
df['2년전_근로자수'] = group['근로자수'].shift(2)

# 증가율 피처
df['자산증가율'] = (df['전년_자산총계'] - df['2년전_자산총계']) / df['2년전_자산총계'].replace(0, np.nan)
df['부채증가율'] = (df['전년_부채총계'] - df['2년전_부채총계']) / df['2년전_부채총계'].replace(0, np.nan)
df['근로자수증가율'] = (df['전년_근로자수'] - df['2년전_근로자수']) / df['2년전_근로자수'].replace(0, np.nan)

# 기존 피처
df['전년_영업이익'] = group['손익계산서_영업이익'].shift(1)
df['전년_매출증가율'] = np.where(df['2년전_매출액'] > 0,
                                (df['전년_매출액'] - df['2년전_매출액']) / df['2년전_매출액'], np.nan)
df['전년_영업이익률'] = np.where(df['전년_매출액'] > 0,
                                df['전년_영업이익'] / df['전년_매출액'], np.nan)

# 업종 평균 대비
industry_avg = df.groupby('업종')[['손익계산서_매출액', '손익계산서_영업이익']].transform('mean')
df['업종평균_매출차이'] = df['손익계산서_매출액'] - industry_avg['손익계산서_매출액']
df['업종평균_이익차이'] = df['손익계산서_영업이익'] - industry_avg['손익계산서_영업이익']
```

```
# 안전 로그 변환
def safe_log(x):
    return np.sign(x) * np.log1p(np.abs(x))

log_cols = ['손익계산서_매출액', '손익계산서_영업이익', '재무상태표_자산총계',
            '재무상태표_부채총계', '재무상태표_자본총계',
            '재무상태표_유동자산', '재무상태표_유동부채']

for col in log_cols:
    df[f'safe_log_{col}'] = safe_log(df[col])

df['순입사자수'] = df['입사자수'].fillna(0) - df['퇴사자수'].fillna(0)
df['입사율'] = df['입사자수'].fillna(0) / df['근로자수'].replace(0, np.nan)
df['퇴사율'] = df['퇴사자수'].fillna(0) / df['근로자수'].replace(0, np.nan)

df['유동비율'] = df['재무상태표_유동자산'] / df['재무상태표_유동부채'].replace(0, np.nan)
df['부채비율'] = df['재무상태표_부채총계'] / df['재무상태표_자본총계'].replace(0, np.nan)
df['자본잉여금비율'] = df['재무상태표_자본잉여금'] / df['재무상태표_자본금'].replace(0, np.nan)
df['safe_log_유동비율'] = safe_log(df['유동비율'])
df['safe_log_부채비율'] = safe_log(df['부채비율'])

df['업종코드'] = LabelEncoder().fit_transform(df['업종'].astype(str))

print("✅ 피처 엔지니어링 완료!")
```


데이터 전처리

- 피쳐 타겟 정의
- 결측치 & 이상치 처리
- 스케일링
- 학습/테스트 데이터 분리

```
# ✅ 4. Train/Test Split
train_df = df[df['기준연도'] <= df['기준연도'].max() - 1].copy()
test_df = df[df['기준연도'] == df['기준연도'].max()].copy()

for dataset in [train_df, test_df]:
    dataset.loc[:, '매출증가율'] = np.where(
        dataset['전년_매출액'] > 0,
        (dataset['손익계산서_매출액'] - dataset['전년_매출액']) / dataset['전년_매출액'],
        np.nan
    )
    dataset.loc[:, '영업이익률'] = np.where(
        dataset['손익계산서_매출액'] > 0,
        dataset['손익계산서_영업이익'] / dataset['손익계산서_매출액'],
        np.nan
    )

features = [
    '전년_매출액', '전년_영업이익', '전년_매출증가율', '전년_영업이익률',
    '자산증가율', '부채증가율', '근로자수증가율',
    'safe_log_손익계산서_매출액', 'safe_log_손익계산서_영업이익',
    'safe_log_재무상태표_자산총계', 'safe_log_재무상태표_부채총계',
    'safe_log_재무상태표_자본총계', '근로자수', '순입사자수', '입사율', '퇴사율',
    '업종평균_매출차이', '업종평균_이익차이',
    'safe_log_유동비율', 'safe_log_부채비율', '자본잉여금비율',
    '업종코드'
]
```

```
target_growth = '매출증가율'
target_profit = '영업이익률'

# ✅ 타겟만 dropna
train_df = train_df.dropna(subset=[target_growth, target_profit])
test_df = test_df.dropna(subset=[target_growth, target_profit])

print(f"✅ 학습용 데이터: {len(train_df)} rows")
print(f"✅ 검증용 데이터: {len(test_df)} rows")

# ✅ 피쳐는 NaN 대체
X_train = train_df[features].fillna(0)
y_growth_train = train_df[target_growth]
y_profit_train = train_df[target_profit]

X_test = test_df[features].fillna(0)
y_growth_test = test_df[target_growth]
y_profit_test = test_df[target_profit]

# ✅ 스케일링
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

모델 학습 검증

```
# ✅ 5. RandomForest 기본 모델 학습
model_growth = RandomForestRegressor(random_state=42, n_estimators=100)
print("🚀 RandomForest 모델 학습 시작 (매출증가율)...")
model_growth.fit(X_train_scaled, y_growth_train)
print("✅ 매출증가율 모델 학습 완료!")

model_profit = RandomForestRegressor(random_state=42, n_estimators=100)
print("🚀 RandomForest 모델 학습 시작 (영업이익률)...")
model_profit.fit(X_train_scaled, y_profit_train)
print("✅ 영업이익률 모델 학습 완료!")

# ✅ 6. 검증
y_growth_pred = model_growth.predict(X_test_scaled)
y_profit_pred = model_profit.predict(X_test_scaled)

rmse_growth = np.sqrt(mean_squared_error(y_growth_test * 100, y_growth_pred * 100))
mae_growth = mean_absolute_error(y_growth_test * 100, y_growth_pred * 100)

rmse_profit = np.sqrt(mean_squared_error(y_profit_test * 100, y_profit_pred * 100))
mae_profit = mean_absolute_error(y_profit_test * 100, y_profit_pred * 100)

print(f"📊 매출증가율 RMSE: {rmse_growth:.2f}%, MAE: {mae_growth:.2f}%")
print(f"📊 영업이익률 RMSE: {rmse_profit:.2f}%, MAE: {mae_profit:.2f}%")
```

모델 학습 및 검증

기업 성장성·수익성 예측 모델 학습 및 검증

모델 성능 비교 결과



해석

1. RandomForest

- 성능 가장 안정적
- 매출증가율 RMSE: 54.36%, MAE: 9.52%
- 영업이익률 RMSE: 77.64%, MAE: 8.24%
- 학습시간은 꽤 걸리지만 결과는 가장 믿을만합니다.

2. XGBoost

- 기대보다 성능 낮음
- RMSE, MAE 모두 RandomForest 대비 2배 이상 높음
- 다만, 학습 시간은 매우 빠름 (약 1초)
- 결론적으로 XGBoost 가 제대로 피팅되지 않은 것으로 보입니다.
- 원인 가능성:
 - 기본 파라미터라서 underfitting
 - 표준화 된 데이터를 사용했는데 scale 관련 issue 가능성
 - 피처 엔지니어링이 RandomForest 에 더 유리한 구조

3. HistGradientBoosting

- 심각한 과대/과소적합 또는 학습 실패
- RMSE 수치가 비정상적으로 큼니다. (7000% 이상)
- 학습 시간은 빠르지만 결과가 너무 나쁩니다.
- 결론: 현재 설정에서는 사용할 수 없는 수준.

모델	타겟	RMSE (%)	MAE (%)	학습 시간 (초)
RandomForest	매출증가율	54.36	9.52	149.35
RandomForest	영업이익률	77.64	8.24	123.81
XGBoost	매출증가율	129.81	20.87	0.73
XGBoost	영업이익률	123.19	14.40	0.72
HistGB	매출증가율	7183.16	654.60	0.23
HistGB	영업이익률	666.85	96.82	0.23

하이퍼파라미터란?



정의

- 머신러닝 모델이 학습을 시작하기 전에 사용자가 설정하는 값
- 모델의 구조와 학습 방식에 큰 영향을 줌
- 학습 중 데이터로부터 자동으로 학습되지 않음!

구분

예시

Model Hyperparameter

RandomForest: n_estimators, max_depth
XGBoost: learning_rate, max_depth, n_estimators

Training Hyperparameter

batch_size, learning_rate, epochs 등

“좋은 데이터가 준비되었다면, 그 다음은 최적의 하이퍼파라미터를 찾는 것이 모델링의 핵심입니다.”

하이퍼파라미터 튜닝

기업 성장성·수익성 예측 모델 학습 및 검증

왜 튜닝이 중요한가?

과소적합 방지

- 복잡도가 너무 낮으면 데이터 패턴을 놓칠 수 있음

과적합 방지

- 너무 복잡하면 훈련 데이터에만 맞춰져 새로운 데이터 예측력 저하

최적의 성능 달성

- 동일한 데이터로도 하이퍼파라미터에 따라 모델 성능이 크게 달라짐

“모델 선택보다 더 중요한 게 하이퍼파라미터 튜닝입니다. 동일한 모델이라도 성능은 천차만별!”

하이퍼파라미터 튜닝

기업 성장성·수익성 예측 모델 학습 및 검증

주요 하이퍼파라미터 예시 (RandomForest)

하이퍼파라미터	설명	기본값
n_estimators	트리 개수 (많을수록 안정적)	100
max_depth	트리 최대 깊이 (과적합 방지)	None
min_samples_split	노드 분할 최소 샘플 수	2
min_samples_leaf	리프 노드 최소 샘플 수	1
max_features	최적의 분할을 위한 피처 수	auto'

"무작정 높게 설정하는 것이 정답은 아닙니다. 복잡성 ↔ 일반화 능력 균형!"

튜닝 방법론

1 그리드 서치 (Grid Search)

- 미리 정의된 값들을 완전 탐색
- 장점: 모든 조합 시도
- 단점: 시간이 많이 소요됨

2 랜덤 서치 (Random Search)

- 임의로 샘플링하여 빠르게 탐색
- 적은 시간으로 좋은 결과 가능

3 베이지안 최적화

- 확률적 모델 기반으로 최적화
- 탐색 효율이 가장 뛰어남 (Optuna 등 사용)

“작게는 Random Search, 크게는 베이지안 최적화를 추천합니다!”

하이퍼파라미터 튜닝

기업 성장성·수익성 예측 모델 학습 및 검증

요약

구분	설명
하이퍼파라미터	모델 성능을 결정짓는 핵심 요인
튜닝 방법	Grid Search, Random Search, Bayesian Optimization
실습	GridSearchCV 으로 최적 파라미터 찾기
기대 효과	과적합 방지, 성능 최적화

하이퍼파라미터 튜닝

기업 성장성·수익성 예측 모델 학습 및 검증

실습

```
# ✅ 13. RandomForestRegressor (Grid Search)
print("\n🚀 RandomForest + Grid Search 시작!")

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10, None],
    'min_samples_split': [2, 5],
}

def grid_search_and_evaluate(X_train, X_test, y_train, y_test, target_name):
    print(f"🔍 Grid Search - {target_name} 시작!")
    grid_search = GridSearchCV(
        RandomForestRegressor(random_state=42),
        param_grid,
        cv=3,
        scoring='neg_mean_squared_error',
        verbose=2
    )
    grid_search.fit(X_train, y_train)

    best_model = grid_search.best_estimator_
    print(f"✅ [GridSearch] {target_name} Best Params: {grid_search.best_params_}")

    predictions = best_model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, predictions))
    mae = mean_absolute_error(y_test, predictions)

    print(f"📊 [GridSearch] {target_name} RMSE: {rmse:.2f}, MAE: {mae:.2f}")
    return best_model

# ✅ Grid Search - 매출증가율
best_growth_model = grid_search_and_evaluate(X_train, X_test, y_growth_train, y_growth_test, "매출증가율")

# ✅ Grid Search - 영업이익률
best_profit_model = grid_search_and_evaluate(X_train, X_test, y_profit_train, y_profit_test, "영업이익률")

print("\n✅ 모든 모델 학습 및 평가 완료!")
```

```
🚀 RandomForest + Grid Search 시작!
🔍 Grid Search - 매출증가율 시작!
Fitting 3 folds for each of 12 candidates, totalling 36 fits
[CV] END .max_depth=5, min_samples_split=2, n_estimators=100; total time= 8.0s
[CV] END .max_depth=5, min_samples_split=2, n_estimators=100; total time= 7.3s
[CV] END .max_depth=5, min_samples_split=2, n_estimators=100; total time= 8.2s
[CV] END .max_depth=5, min_samples_split=2, n_estimators=200; total time= 15.4s
[CV] END .max_depth=5, min_samples_split=2, n_estimators=200; total time= 15.3s
[CV] END .max_depth=5, min_samples_split=2, n_estimators=200; total time= 15.4s
[CV] END .max_depth=5, min_samples_split=5, n_estimators=100; total time= 7.8s
[CV] END .max_depth=5, min_samples_split=5, n_estimators=100; total time= 7.6s
[CV] END .max_depth=5, min_samples_split=5, n_estimators=100; total time= 8.2s
[CV] END .max_depth=5, min_samples_split=5, n_estimators=200; total time= 15.6s
[CV] END .max_depth=5, min_samples_split=5, n_estimators=200; total time= 15.3s
[CV] END .max_depth=5, min_samples_split=5, n_estimators=200; total time= 15.3s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time= 15.9s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time= 14.8s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time= 14.9s
[CV] END max_depth=10, min_samples_split=2, n_estimators=200; total time= 30.0s
[CV] END max_depth=10, min_samples_split=2, n_estimators=200; total time= 29.8s
[CV] END max_depth=10, min_samples_split=2, n_estimators=200; total time= 30.5s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time= 15.4s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time= 15.2s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time= 14.9s
[CV] END max_depth=10, min_samples_split=5, n_estimators=200; total time= 30.2s
[CV] END max_depth=10, min_samples_split=5, n_estimators=200; total time= 29.5s
[CV] END max_depth=10, min_samples_split=5, n_estimators=200; total time= 30.1s
[CV] END max_depth=None, min_samples_split=2, n_estimators=100; total time= 30.4s
[CV] END max_depth=None, min_samples_split=2, n_estimators=100; total time= 27.6s
[CV] END max_depth=None, min_samples_split=2, n_estimators=100; total time= 29.0s
[CV] END max_depth=None, min_samples_split=2, n_estimators=200; total time= 1.0min
[CV] END max_depth=None, min_samples_split=2, n_estimators=200; total time= 54.2s
[CV] END max_depth=None, min_samples_split=2, n_estimators=200; total time= 58.3s
[CV] END max_depth=None, min_samples_split=5, n_estimators=100; total time= 28.5s
[CV] END max_depth=None, min_samples_split=5, n_estimators=100; total time= 25.0s
[CV] END max_depth=None, min_samples_split=5, n_estimators=100; total time= 27.2s
[CV] END max_depth=None, min_samples_split=5, n_estimators=200; total time= 56.6s
[CV] END max_depth=None, min_samples_split=5, n_estimators=200; total time= 51.0s
[CV] END max_depth=None, min_samples_split=5, n_estimators=200; total time= 54.2s
✅ [GridSearch] 매출증가율 Best Params: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}
📊 [GridSearch] 매출증가율 RMSE: 97.37, MAE: 1.78
```


예측 결과 생성

기업 성장성·수익성 예측 모델 학습 및 검증

✅ 예측값 계산 & 저장

```
# ✅ 7. 미래 예측 및 저장
def predict_future(model_growth, model_profit, scaler, original_df, years_ahead=3):
    prediction_results = []
    print("🌀 미래 예측 진행 중...")
    biz_total = train_df['사업자번호'].nunique()
    for idx, (biz_no, group) in enumerate(train_df.groupby('사업자번호'), start=1):
        if idx % 100 == 0:
            print(f"➡ {idx}/{biz_total} 기업 예측 완료")
        group = group.sort_values('기준연도')
        if len(group) < 2:
            continue

        last_row = group.iloc[-1].copy()

        for i in range(1, years_ahead + 1):
            next_year = last_row['기준연도'] + 1
            input_features = pd.DataFrame([{'feat': last_row.get(feat, 0) for feat in features}])
            input_scaled = scaler.transform(input_features)

            predicted_growth = model_growth.predict(input_scaled)[0]
            predicted_profit = model_profit.predict(input_scaled)[0]

            prediction_results.append({
                '사업자번호': biz_no,
                '기업명': last_row['기업명'],
                '기준연도': next_year,
                '예측_매출증가율': predicted_growth * 100,
                '예측_영업이익률': predicted_profit * 100
            })

            last_row['기준연도'] = next_year
            last_row['손익계산서_매출액'] *= (1 + predicted_growth)
            last_row['매출증가율'] = predicted_growth
            last_row['영업이익률'] = predicted_profit

    prediction_df = pd.DataFrame(prediction_results)
    print(f"✅ 미래 예측 완료! 총 {len(prediction_df)}건")
    return prediction_df
```

```
future_predictions = predict_future(model_growth, model_profit, scaler, train_df)

if not future_predictions.empty:
    con = duckdb.connect(database='company_data.duckdb', read_only=False)
    con.execute("DROP TABLE IF EXISTS prediction_results")
    con.execute("CREATE TABLE prediction_results AS SELECT * FROM future_predictions")
    con.close()
    print("✅ 예측 결과 DuckDB 저장 완료!")
else:
    print("⚠️ 예측 결과가 비어 있습니다. 저장을 건너웁니다.")
```

실습

```
import streamlit as st
import pandas as pd
import duckdb
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm

# Streamlit 페이지 설정
st.set_page_config(page_title='기업 재무 분석 및 예측 대시보드', layout='wide')
st.title('📊 기업 재무 분석 및 예측 대시보드')
st.markdown('사업자번호 또는 기업명을 입력하여 기업을 검색하고 성장성 및 수익성, 그리고 예측 결과를 확인하세요.')

# 폰트 설정
font_dirs = ['/usr/share/fonts/truetype/nanum/']
font_files = fm.findSystemFonts(fontpaths=font_dirs)
for font_file in font_files:
    fm.fontManager.addfont(font_file)

nanum_font = fm.FontProperties(fname=font_files[0]).get_name()
plt.rcParams['font.family'] = nanum_font
plt.rcParams['axes.unicode_minus'] = False

# ✅ DuckDB 연결 및 데이터 로딩
@st.cache_data
def load_data():
    con = duckdb.connect(database='company_data.duckdb', read_only=False)
    company_df = con.execute("SELECT * FROM company_data").df()
    prediction_df = con.execute("SELECT * FROM prediction_results").df()
    con.close()
    return company_df, prediction_df

company_data, prediction_data = load_data()

# ✅ 데이터가 존재하는 기업만 필터링
valid_biz_no = company_data.dropna(subset=['손익계산서_매출액', '손익계산서_영업이익']).사업자번호.unique()

company_data_filtered = company_data[company_data['사업자번호'].isin(valid_biz_no)]
prediction_data_filtered = prediction_data[prediction_data['사업자번호'].isin(valid_biz_no)]

# ✅ 사용자 입력: 사업자번호 또는 기업명 검색
search_input = st.text_input('🔍 사업자번호 또는 기업명을 입력하세요')
```


실습

```
if search_input:
    filtered_data = company_data_filtered[
        company_data_filtered['사업자번호'].astype(str).str.contains(search_input) |
        company_data_filtered['기업명'].str.contains(search_input, na=False)
    ].drop_duplicates(subset=['사업자번호'])

if not filtered_data.empty:
    st.write(f"🔍 {len(filtered_data)}개의 기업이 검색되었습니다.")
    selected_company_name = st.selectbox('✅ 분석할 기업명을 선택하세요:', filtered_data['기업명'].unique())

    selected_biz_no = filtered_data[filtered_data['기업명'] == selected_company_name]['사업자번호'].values[0]

    company_df = company_data_filtered[company_data_filtered['사업자번호'] == selected_biz_no].sort_values('기준연도')
    prediction_df = prediction_data_filtered[prediction_data_filtered['사업자번호'] == selected_biz_no].sort_values('기준연도')

    selected_industry = company_df['업종'].iloc[0] if not company_df.empty else '정보 없음'
    st.markdown(f"### 📌 선택된 기업: {selected_company_name} (사업자번호: {selected_biz_no})")
    st.markdown(f"#### 💎 소속 업종: {selected_industry}")

    industry_df = company_data_filtered[company_data_filtered['업종'] == selected_industry].groupby('기준연도').mean(numeric_only=True).reset_index()

# ✅ 차트
col1, col2 = st.columns(2)

with col1:
    st.subheader('📈 매출 증가율 추이 (과거 + 예측)')
    fig, ax = plt.subplots(figsize=(6, 4))
    ax.plot(company_df['기준연도'], company_df['매출증가율'], marker='o', label='과거 데이터', color='skyblue')
    ax.scatter(prediction_df['기준연도'], prediction_df['예측_매출증가율'], color='orange', label='예측', zorder=5)
    ax.set_xlabel('기준연도')
    ax.set_ylabel('매출 증가율 (%)')
    ax.legend()
    ax.grid(True)
    st.pyplot(fig)

with col2:
    st.subheader('📈 영업이익률 추이 (과거 + 예측)')
    fig, ax = plt.subplots(figsize=(6, 4))
    ax.plot(company_df['기준연도'], company_df['영업이익률'], marker='o', label='과거 데이터', color='salmon')
    ax.scatter(prediction_df['기준연도'], prediction_df['예측_영업이익률'], color='orange', label='예측', zorder=5)
    ax.set_xlabel('기준연도')
    ax.set_ylabel('영업이익률 (%)')
    ax.legend()
    ax.grid(True)
    st.pyplot(fig)
```

실습

```
with col3:
    st.subheader('💰 매출액 추이 (과거 + 예측)')
    fig, ax = plt.subplots(figsize=(6, 4))
    ax.plot(company_df['기준연도'], company_df['손익계산서_매출액'], marker='o', label='과거 매출액', color='green')

    # ✅ 예측 매출액 계산
    if not prediction_df.empty and '예측_매출증가율' in prediction_df.columns:
        future_years = prediction_df['기준연도'].values
        # 과거 마지막 매출액
        last_sales = company_df['손익계산서_매출액'].iloc[-1]
        # 누적 예측 매출액 계산
        predicted_sales = [last_sales]
        for growth_rate in prediction_df['예측_매출증가율']:
            next_sales = predicted_sales[-1] * (1 + growth_rate / 100)
            predicted_sales.append(next_sales)
        predicted_sales.pop(0) # 첫 값 제거 (last_sales 중복)

        ax.scatter(future_years, predicted_sales, color='orange', label='예측 매출액', zorder=5)

    ax.set_xlabel('기준연도')
    ax.set_ylabel('매출액')
    ax.legend()
    ax.grid(True)
    st.pyplot(fig)
```

실습

```
with col4:
    st.subheader('📊 영업이익 추이 (과거 + 예측)')
    fig, ax = plt.subplots(figsize=(6, 4))
    ax.plot(company_df['기준연도'], company_df['손익계산서_영업이익'], marker='o', label='과거 영업이익', color='purple')

    # ✅ 예측 영업이익 계산
    if not prediction_df.empty and '예측_영업이익률' in prediction_df.columns:
        future_years = prediction_df['기준연도'].values
        # 과거 마지막 매출액 및 영업이익률
        last_sales = company_df['손익계산서_매출액'].iloc[-1]
        predicted_sales = [last_sales]
        last_profit_rate = company_df['영업이익률'].iloc[-1]

        predicted_profits = []
        for growth_rate, profit_rate in zip(prediction_df['예측_매출증가율'], prediction_df['예측_영업이익률']):
            # 매출액 누적 예측
            next_sales = predicted_sales[-1] * (1 + growth_rate / 100)
            predicted_sales.append(next_sales)

            # 영업이익 = 매출액 * 영업이익률
            next_profit = next_sales * (profit_rate / 100)
            predicted_profits.append(next_profit)

        predicted_sales.pop(0) # 첫 번째 중복 제거
        ax.scatter(future_years, predicted_profits, color='orange', label='예측 영업이익', zorder=5)

    ax.set_xlabel('기준연도')
    ax.set_ylabel('영업이익')
    ax.legend()
    ax.grid(True)
    st.pyplot(fig)
    # ✅ 데이터 테이블 표시
    st.subheader('📄 기업 데이터 테이블 (과거 + 예측)')
    combined_df = pd.concat([company_df, prediction_df], sort=False)
    st.dataframe(combined_df)

    st.success('✅ 대시보드가 성공적으로 로드되었습니다!')
else:
    st.warning("검색 결과가 없습니다. 다시 입력해주세요.")
else:
    st.info("사업자번호 또는 기업명을 입력해주세요.")
```


결과 화면



기업 재무 분석 및 예측 대시보드

사업자번호 또는 기업명을 입력하여 기업을 검색하고 성장성 및 수익성, 그리고 예측 결과를 확인하세요.

🔍 사업자번호 또는 기업명을 입력하세요

아

🔍 524개의 기업이 검색되었습니다.

✅ 분석할 기업명을 선택하세요:

깔끔하고정돈된세아오일공업

⋮



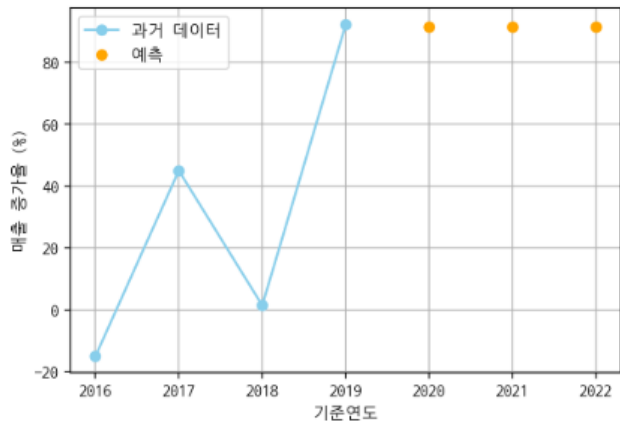
선택된 기업: 깔끔하고정돈된세아오일공업 (사업자번호: 1813251308)



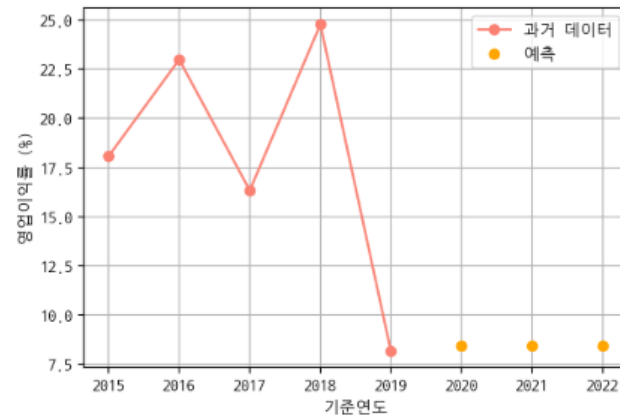
소속 업종: 경영 컨설팅업



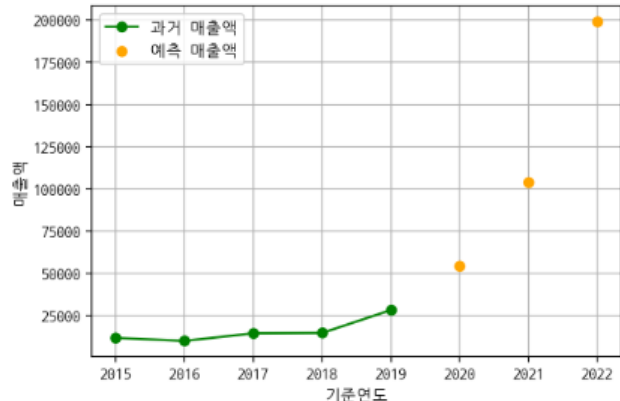
매출 증가율 추이 (과거 + 예측)



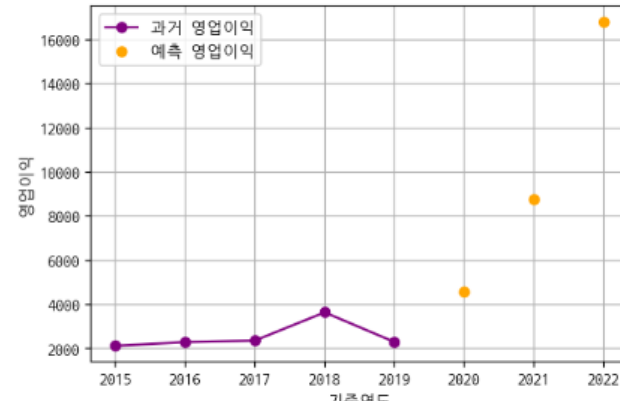
영업이익률 추이 (과거 + 예측)



매출액 추이 (과거 + 예측)



영업이익 추이 (과거 + 예측)



기업 데이터 테이블 (과거 + 예측)

	기업명	사업자번호	법인번호	기준연도	상태	손익계산서_매출액	손익계산서_매출원가	손익계산서_매출
6058	깔끔하고정돈된세아오일공	1813251308	5889111611010	2015	정상	11788	None	
6059	깔끔하고정돈된세아오일공	1813251308	5889111611010	2016	정상	10012	None	
6060	깔끔하고정돈된세아오일공	1813251308	5889111611010	2017	정상	14518	5664	
6061	깔끔하고정돈된세아오일공	1813251308	5889111611010	2018	정상	14726	2132	
6062	깔끔하고정돈된세아오일공	1813251308	5889111611010	2019	정상	28300	9183	

1 우리는 무엇을 했나요?

- ✓ 기업 재무 데이터를 활용해 미래 성장성과 수익성을 예측했습니다.
- ✓ 다양한 모델을 비교하여 최적의 모델을 선택했습니다.
- ✓ 예측 결과를 대시보드로 시각화하여 직관적으로 분석했습니다.

2 우리는 무엇을 배웠나요?

- 📁 데이터 준비
 - 기업 재무, 인사 데이터 수집 및 정리
 - DuckDB 기반 데이터 관리
- 🧩 피처 엔지니어링
 - 성장성, 수익성, 재무 비율, 인사 지표 활용
 - 시계열 데이터의 전년 대비 변화량 적용
- 🧠 모델링 & 평가
 - RandomForest, XGBoost, HistGradientBoosting 모델 비교
 - RMSE, MAE 지표로 모델 성능 평가
- 🖥️ 대시보드 구현
 - Streamlit 기반 대시보드
 - 과거 + 예측 매출액, 영업이익, 성장성 지표 시각화

3 앞으로 우리는 무엇을 할 수 있나요?

- 🔥 고도화
 - 하이퍼파라미터 튜닝으로 예측 정확도 향상
 - Cross-validation 및 예측 신뢰 구간 추가
- 📊 확장
 - 업종별 평균 대비 성과 분석
 - 기업 간 비교 분석 기능 추가
- 🔄 자동화
 - 실시간 데이터 업데이트 및 재학습 파이프라인
 - 대시보드 자동 Refresh
- 📦 실전 적용
 - 실제 투자/경영 판단 보조 도구로 활용
 - 예측 결과 리포트 자동 생성

- ㄱ ㅡ -