



AI-First Business Financial Analysis Workflow - Code Pipeline Breakdown

The following is an overview of the workflow structure through the code sections:



PIPELINE ORCHESTRATOR: `mvp_forecast2.py`

Lines 85-117: Main orchestration function

```
def main():
    # Run webprep first to prepare financial data files
    if run_webprep():
        # Run finscore to analyze the gathered data
        if run_finscore():
            # Run SWOT analysis copy
            if run_copy_swot():
                # Run file zipping
                if run_files_zip():
```

Key Functions:

- `run_webprep()` : Lines 5-24 - Subprocess call to `webprep.py`
- `run_finscore()` : Lines 25-43 - Subprocess call to `finscore.py`
- `run_copy_swot()` : Lines 45-63 - Subprocess call to `copy_swot.py`
- `run_files_zip()` : Lines 65-83 - Subprocess call to `files_zip.py`



STAGE 1: Data Preparation - `webprep.py`

Lines 4-31: File copying and renaming

```
def copy_and_rename_files():
    file_mapping = {
        "file1.txt": "financial_report.txt",
        "file2.txt": "balance_sheet.txt",
        "file3.txt": "cashflow.txt"
    }
```

Purpose:

- Copies raw web files from `/web_files/` directory
 - Renames them to standardized financial statement names
 - Prepares data for downstream analysis
-



STAGE 2: Financial Analysis Engine - `finscore.py`

Lines 149-200: Main scoring orchestration

```
def main():
    if run_financial_analysis():
        final_score = run_score_parse()
        swot_score = run_swot_analysis()
        sm3_score = run_3sm_analysis()
        # Calculate FINSCORE average
        finscore = sum(scores) / len(scores)
        run_3sm4_analysis(finscore)
```

Key Components:

1. `run_financial_analysis()` : Lines 5-20 - Calls `fin_analysis.py`
 2. `run_score_parse()` : Lines 22-48 - Calls `score_parse.py` , extracts final score
 3. `run_swot_analysis()` : Lines 50-93 - Orchestrates SWOT pipeline
 4. `run_3sm_analysis()` : Lines 95-125 - Calls `3sm3.py` forecasting
 5. `run_3sm4_analysis()` : Lines 127-147 - Calls `3sm4.py` with FINSCORE
-



STAGE 3: Core Financial Analysis - `fin_analysis.py`

Lines 5-26: Multi-script orchestration

```
scripts = [
    '3sm.py', 'dcf.py', 'valuation.py', 'ratio_analysis.py',
    'ar_metrics.py', 'op_margin.py', '13w_cashflow.py', 'payback.py',
    'cashcc.py', 'Horizontal_fin.py', 'Vertical_fin.py', 'nvp.py',
    'trend_fin.py', 'dupont.py', 'valuation.py', 'credit1.py',
    'cagr_fin.py', 'Eva.py'
]
```

Purpose:

- Runs 18 different financial analysis scripts
 - Aggregates all outputs into `financial_analysis_report.txt`
 - Comprehensive ratio analysis and valuation metrics
-

STAGE 4: Score Parsing & Interpretation - `score_parse.py`

Lines 1-50: Financial terminology dictionaries

```
profitability_terms = {'profit', 'margin', 'return', 'roi', ...}
liquidity_terms = {'cash', 'liquid', 'current ratio', ...}
solvency_terms = {'debt', 'equity', 'leverage', ...}
# ... 6 more term categories
```

Purpose:

- Defines 7 categories of financial terminology (400+ terms)
 - Parses financial analysis output for scoring
 - Extracts "Overall Average Score" using regex pattern matching
-

STAGE 5: AI/ML Forecasting - `3sm3.py`

Lines 6-20: Financial model initialization

```
class FinancialModel:
    def __init__(self, income_data: str, balance_data: str, cashflow_data:
str):
    self.income_metrics = self._parse_income_data(income_data)
    self.balance_metrics = self._parse_balance_data(balance_data)
    self.cashflow_metrics = self._parse_cashflow_data(cashflow_data)
    self.hw_revenue_forecast = None # Holt-Winters forecast
    self.hw_costs_forecast = None
    self.hw_opex_forecast = None
```

Key Features:

- **Holt-Winters Exponential Smoothing** for time series forecasting
- Parses income statements, balance sheets, cash flow statements
- Multi-year revenue, cost, and OPEX projections

- Calculates overall company score
-

STAGE 6: SWOT Analysis Suite

Three components orchestrated by `finscore.py` :

1. `s-w-o-t.py` : Generates SWOT analysis content
2. `swot_nat.py` : Natural language processing (60s timeout)
3. `swot_score.py` : Sentiment analysis scoring

`swot_score.py` **Key Logic:**

```
weights = {  
    'Strengths': 0.35,      # 35%  
    'Weaknesses': 0.25,    # 25%  
    'Opportunities': 0.20, # 20%  
    'Threats': 0.20        # 20%  
}
```

STAGE 7: SWOT File Management - `copy_swot.py`

Lines 8-18: File organization

```
source_path = os.path.join(current_dir, "swot_nat.txt")  
dest_path = os.path.join(current_dir, "analysis_outputs", "swot_nat.txt")  
shutil.copy2(source_path, dest_path)
```

Purpose:

- Creates `analysis_outputs/` directory
 - Copies SWOT analysis results for packaging
-

STAGE 8: Output Packaging - `files_zip.py`

Lines 4-24: Zip file creation

```
def zip_analysis_outputs():
    source_dir = 'analysis_outputs'
    output_zip = 'analysis_outputs.zip'
    with zipfile.ZipFile(output_zip, 'w', zipfile.ZIP_DEFLATED) as zipf:
        for root, dirs, files in os.walk(source_dir):
            for file in files:
                zipf.write(file_path, arcname)
```

ACTUAL CODE DEPENDENCY FLOW

```
mvp_forecast2.py (MAIN)
├─ webprep.py (Data Prep)
├─ finscore.py (Analysis Engine)
│   ├─ fin_analysis.py (Core Analysis)
│   │   └─ 18 financial scripts
│   ├─ score_parse.py (Score Extraction)
│   ├─ 3sm3.py (Forecasting Model)
│   ├─ SWOT Suite:
│   │   ├─ s-w-o-t.py
│   │   ├─ swot_nat.py
│   │   └─ swot_score.py
│   └─ 3sm4.py (Enhanced Forecasting)
├─ copy_swot.py (File Mgmt)
└─ files_zip.py (Packaging)
```

TECHNICAL PIPELINE CHARACTERISTICS

- **Orchestration:** Subprocess-based modular architecture
- **Error Handling:** Try/catch blocks with detailed error messages
- **Data Flow:** File-based communication between components
- **Scoring Integration:** Regex pattern matching for score extraction
- **Timeout Management:** 60-second timeout for NLP operations
- **File Organization:** Structured output directory with zip packaging

This workflow leverages **advanced data science orchestration** with robust error handling, modular design, and comprehensive financial analysis capabilities.