Financial Insights Dashboard and Forecasting Model

This project builds a comprehensive Financial Insights Dashboard and a Financial Scoring Model to analyze family-level financial health and forecast monthly expenses. It also includes an API and an optional interactive web application for users to interact with the scoring model and visualizations.

Features

1. **Data Preprocessing**
    - Handles missing data through removal or imputation.
    - Aggregates transaction data to a monthly level for analysis and forecasting.
    - Indexes the dataset by transaction date for time series analysis.
    - Ensures data stationarity for effective time series modeling.
2. **Financial Insights & Scoring**
    - Analyzes correlations between key financial metrics:
        - Positive: Income vs. Savings.
        - Negative: Loan Payments vs. Income.
    - Visualizes insights like:
        - Spending distribution across categories.
        - Family-wise financial scores.
        - Member-wise spending trends.
    - Computes financial health scores (0–100) using:
        - Savings-to-Income Ratio
        - Monthly Expenses-to-Income Ratio
        - Loan Payments-to-Income Ratio
        - Credit Card Spending Trends
        - Category Spending Distribution
        - Percentage of Financial Goals Met
3. **Forecasting Models**
    - ARIMA Model:
        - Tests stationarity using the ADF Test.
        - Forecasts monthly expenses using historical trends.

- SARIMAX Model:
    - Incorporates exogenous variables (Income and Savings) for predictions.
    - Accounts for seasonality in monthly data.

4. **API and Web Interface**

(Note: I am no used to with FLASK app and DASH app. I need to learn these things. So that’s why I kept left here.)

Files in the Repository

1. forecasting_model.py: Main script for training the SARIMAX model, generating forecasts, and computing financial scores.
2. data.csv: Sample dataset for model training and testing, with columns like Monthly Expenses, Income, and Savings.
3. requirements.txt: Python dependencies for the project.
4. README.md: Setup instructions and project details.

Dependencies

To run this project, you will need the following Python libraries:

- pandas: Data manipulation and analysis.
- numpy: Numerical operations.
- matplotlib: Visualization of results.
- seaborn: Advanced data visualizations.
- statsmodels: Building and evaluating ARIMA/SARIMAX models.
- scikit-learn: Data preprocessing and evaluation.
- Flask or FastAPI: API development.
- Streamlit or Dash (optional): Interactive web application.

Install all dependencies using:

bash

Copy code

```
pip install -r requirements.txt
```

Usage

1. Data Preprocessing
    - Load and clean the financial dataset.
    - Aggregate data to a monthly level and index it by date.
    - Check for stationarity and preprocess as needed.
2. Train and Evaluate Models
    - ARIMA: Fits the model to historical data to predict monthly expenses.
    - SARIMAX: Incorporates exogenous regressors (Income and Savings) for improved accuracy and accounts for seasonality.
3. Financial Scoring
    - Compute family financial scores based on key metrics:
        - Savings-to-Income Ratio
        - Monthly Expenses-to-Income Ratio
        - Loan Payments-to-Income Ratio
    - Normalize and weight scores to produce a final score on a 0–100 scale.
4. Visualizations
    - Use matplotlib and seaborn to generate:
        - Spending distribution charts.
        - Family-wise financial scores.
        - Member-wise spending trends.
5. Forecasting
    - Generate forecasts for the next 12 months using SARIMAX.
    - Visualize historical and forecasted data with confidence intervals.
6. API Deployment
    - Deploy the scoring model with Flask/FastAPI.
    - Input: Family-level transaction data.
    - Output: Financial score and key insights.

Results

The SARIMAX model and ARIMA model predicts the monthly expenses for the next month, incorporating historical data and external factors. The output of the forecast will show the predicted values along with confidence intervals, which can be visualized for easier interpretation.