Project Report

# 1.  Introduction

## Project overviews

Unemployed Beneficiary Insurance Forecasting is a predictive analytics initiative designed to estimate the future number of individuals applying for and receiving unemployment insurance benefits. By leveraging historical data and advanced time series modeling, the project aims to empower government agencies, policymakers, and insurance providers with actionable forecasts for better resource allocation and planning.

## Objectives

* Accurately forecast the number of unemployment insurance beneficiaries for upcoming periods.
* Enable data-driven decision-making for budget allocation and policy formulation.
* Develop a user-friendly web application for real-time beneficiary forecasting.

# 2. Project Initialization and Planning Phase

## 2.1. Define Problem Statement

Government agencies and insurance providers face significant challenges in anticipating the demand for unemployment insurance benefits. Fluctuations in economic conditions, labor market trends, and demographic changes make it difficult to allocate resources efficiently and respond proactively to periods of high unemployment.

## 2.2. Project Proposal (Proposed Solution)

This project proposes a data-driven solution using time series forecasting models—such as ARIMA, SARIMA, AutoReg, VAR, and Prophet—to predict future beneficiary counts. The solution includes:

* Data collection and preprocessing from official government sources.
* Model development, evaluation, and optimization.
* Deployment of an interactive web application for stakeholders to access forecasts.

## 2.3. Initial Project Planning

* **Team**: 4 members (roles: data engineer, data scientist, application developer)
* **Timeline**: 2 weeks
* **Milestones**:
  + Data collection and cleaning
  + Exploratory data analysis
  + Model development and validation
  + Model optimization and selection
  + Web application deployment and documentation

# 3. Data Collection and Preprocessing Phase

## 3.1. Data Collection Plan and Raw Data Sources Identified

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Source Name** | **Description** | **Location/URL** | **Format** | **Size** | **Access Permissions** |
| Dataset 1 | Monthly records of unemployment insurance beneficiaries, benefit amounts, regions, and counties in New York State. | <https://data.world/data-ny-gov/xbjp-8sra> | CSV | ~2 MB | Public |

## 3.2. Data Quality Report

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Source** | **Data Quality Issue** | **Severity** | **Resolution Plan** |
| Dataset | Presence of null values in some columns | Moderate | Used df.isnull().sum() to identify missing values. Applied df.dropna(inplace=True) to remove rows with nulls. |
| Dataset | Potential duplicate records | Low | Checked with df.duplicated().sum(). No duplicates found, so no further action required. |
| Dataset | Inconsistent region/county names | Low | Used df['region'].value\_counts() and standardized names if inconsistencies were found. |
| Dataset | Data type inconsistencies | Low | Verified with df.info(). Converted columns to correct data types as needed. |

## 3.3. Data Preprocessing

|  |  |
| --- | --- |
| **Step** | **Code Example / Description** |
| Loading Data | df = pd.read\_csv('unemployment-insurance-beneficiaries-and-benefit-amounts-paid-beginning-2001-1.csv') |
| Handling Missing Data | df.isna().sum() to check for nulls; df.dropna(inplace=True) to remove missing values. |
| Data Transformation | df.columns = df.columns.str.strip() to clean column names. |
| Feature Engineering | df['Beneficiaries\_diff'] = df['Beneficiaries'].diff() to create a new feature capturing monthly changes. |
| Save Processed Data | df.to\_csv('processed\_unemployment\_data.csv', index=False) |

# 4. Model Development Phase

## 4.1. Model Selection Report

|  |  |
| --- | --- |
| **Model** | **Description** |
| ARIMA | Captures autocorrelation in univariate, stationary time series using autoregressive and moving average terms. |
| SARIMA | Extends ARIMA to model both seasonal and non-seasonal patterns. |
| AutoReg | Uses a specified number of past observations for univariate forecasting. |
| VAR | Multivariate model capturing interdependencies between multiple time-dependent variables. |
| Prophet | Additive regression model by Facebook, handles strong seasonal effects and trends, robust to missing data and outliers. |

## 4.2. Initial Model Training Code, Model Validation and Evaluation Report

**Sample Model Training Code (ARIMA):**

from statsmodels.tsa.arima.model import ARIMA

model\_arima = ARIMA(train['Beneficiaries\_diff'].dropna(), order=(5,0,0))

model\_arima\_fit = model\_arima.fit()

print(model\_arima\_fit.summary())

**Validation & Evaluation Metrics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Test MSE** | **Test MAE** | **Test R2** |
| ARIMA | 102,763,733.35 | 5,691.37 | -8.18e-05 |
| SARIMA | 103,545,015.68 | 5,862.50 | NA |
| AutoReg | 102,771,796.73 | 5,862.50 | NA |
| Prophet | 57,301,995.56 | 3,522.24 | -0.1636 |

# 5. Model Optimization and Tuning Phase

## 5.1. Tuning Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Tuned Hyperparameters** | **Optimal Values** | **Description** |
| ARIMA | p, d, q | (5, 0, 0) | p: AR terms, d: differencing, q: MA terms |
| SARIMA | p, d, q, seasonal\_order | (5, 0, 0), (0,1,2,3) | Adds seasonal AR, differencing, MA, and period |
| AutoReg | lags | 10 | Number of previous time steps used for forecasting |
| Prophet | NA | Default | Used default settings; no tuning performed |

## 5.2. Final Model Selection Justification

|  |  |
| --- | --- |
| **Final Model** | **Reasoning** |
| Prophet | Achieved lowest MSE and MAE, handled trends and seasonality automatically, and required minimal tuning. |

# 6. Results

## 6.1. Output Screenshots

* **VSCode:** Screenshots of the Flask code.
* **Web Application:** Screenshots of the deployed Flask app showing user input and forecast output.

# 7. Advantages & Disadvantages

**Advantages:**

* Accurate forecasting for policy and budget planning.
* Handles trends and seasonality automatically (Prophet).
* User-friendly web interface for non-technical users.
* Modular workflow for easy updates and maintenance.

**Disadvantages:**

* Dependent on quality and granularity of historical data.
* Limited to time series models; may not capture sudden economic shocks.
* Prophet’s R2 score remains negative, indicating room for further improvement.

# 8. Conclusion

The Unemployed Beneficiary Insurance Forecasting project delivers a robust, end-to-end solution for forecasting unemployment insurance demand. By combining rigorous data preprocessing, multiple time series models, and a user-friendly web app, the project equips stakeholders with actionable insights for resource allocation and policy decisions. Prophet emerged as the optimal model, balancing accuracy and automation.

# 9. Future Scope

* Integrate additional economic indicators (e.g., unemployment rate, GDP).
* Expand model support to include deep learning (e.g., LSTM, GRU).
* Enhance the web app with interactive dashboards and export features.
* Automate data updates for real-time forecasting.
* Support for multiple states or regions.

# 10. Appendix

## 10.1. Source Code

* All scripts for data preprocessing, modeling, evaluation, and web application are included in the project repository.

## 10.2. GitHub & Project Demo Link

* <https://github.com/Aditya-Sachan-Git/Unemployed-Insurance-Beneficiary-Forecasting>
* <https://drive.google.com/file/d/14Mdqhh-4zJq4Y-wkg4LsMYVQzd-ZADsq/view?usp=sharing>