Final Project Report Template

1. Introduction

Unemployed Insurance Beneficiary (UIB) forecasting involves predicting the number of individuals who will receive unemployment insurance benefits, enabling governments and policymakers to make informed decisions about labor market support. Accurate UIB forecasting can help mitigate the economic and social impacts of unemployment, ensuring effective allocation of resources and support for those in need.

1.1. Project overviews

Unemployed Insurance Beneficiary (UIB) forecasting provides insights into future unemployment trends, enabling proactive policy interventions and resource allocation. It uses machine learning and time series techniques to predict the number of unemployed individuals eligible for benefits, enabling proactive policy decisions and resource allocation. By analyzing historical data and trends, UIB forecasting helps governments and policymakers anticipate and prepare for future unemployment fluctuations. UIB forecasting models can predict the number of beneficiaries, facilitating data-driven decision-making.

1.2. Objectives

The objective of Unemployed Insurance Beneficiary (UIB) forecasting is:

- To accurately predict the number of individuals who will receive unemployment insurance benefits.
- Make informed decisions about resource allocation and policy interventions
- Proactively manage unemployment rates and mitigate the impact of economic downturns
- Ensure timely and effective support for unemployed individuals and their families.

This enables governments and policymakers to make informed decisions about labor market support, resource allocation, and policy interventions.

2. Project Initialization and Planning Phase

2.1. Define Problem Statement

As an unemployment insurance provider, I struggle to accurately predict the number of beneficiaries who will file claims each month. This lack of foresight leads to inefficient resource allocation, delayed payments, and a poor overall experience for our beneficiaries. Furthermore, manual forecasting methods are time-consuming and prone to errors, taking away from the time and resources we could be dedicating to supporting our beneficiaries. The problem of Unemployed Insurance Beneficiary (UIB) forecasting involves accurately predicting the dynamic and volatile number of individuals who will receive unemployment insurance benefits. This is challenging due to the complex interplay of economic, social, and demographic factors that influence unemployment rates.

2.2. Project Proposal (Proposed Solution)

The proposed solution approach involves developing a predictive analytics model leveraging machine learning algorithms, historical data, and economic indicators to accurately forecast the number of unemployed insurance beneficiaries. It includes predictive analytics, machine learning algorithms, data integration, automated forecasting, and real-time reporting and visualization. I need a reliable and accurate forecasting solution that can help me better anticipate and prepare for the needs of our beneficiaries. By improving our forecasting capabilities, we can reduce delays, improve the beneficiary experience, and optimize our resources to better serve those in need.

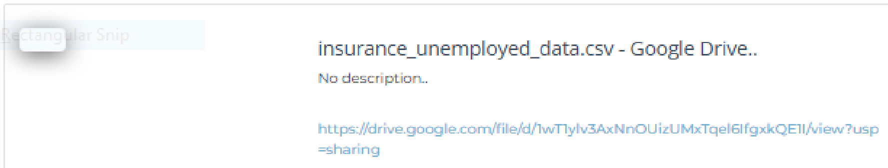
2.3. Initial Project Planning

- **Data Collection and Preprocessing**: Understanding &loading data, Data Preparation, checking Null values & duplicates.
- **Visualizing and Analyzing the Data**: Descriptive Analysis, Exploratory Data Analysis.
- **Model Building**: Splitting Dataset into Train and Test sets
- **Model Development**: Training the model Evaluating the model
- **Model tuning and testing**: Testing the model & Evaluating the metrics. Performance testing.
- **Web Integration and Deployment**: Building HTML templates. Local deployment

3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified

In this project we have used .csv data. This data is downloaded from kaggle.com.



Raw Data Sources Identified: 'Year',Month','Region','County','Beneficiaries','Benefit Amount (Dollars).Later added 'Beneficiaries-diff'. 'Year-Month"-changed to 'date'.This data appears to be related to unemployed insurance beneficiary forecasting, with various variables capturing characteristics of beneficiary count. The datasets likely aims to support classification , prediction,time series,reduce delays, economic indicators, of unemployed Insurance, enabling Government Agency ,Insurance Provider, A Research Institute Professionals to make informed decisions.

3.2. Data Quality Report

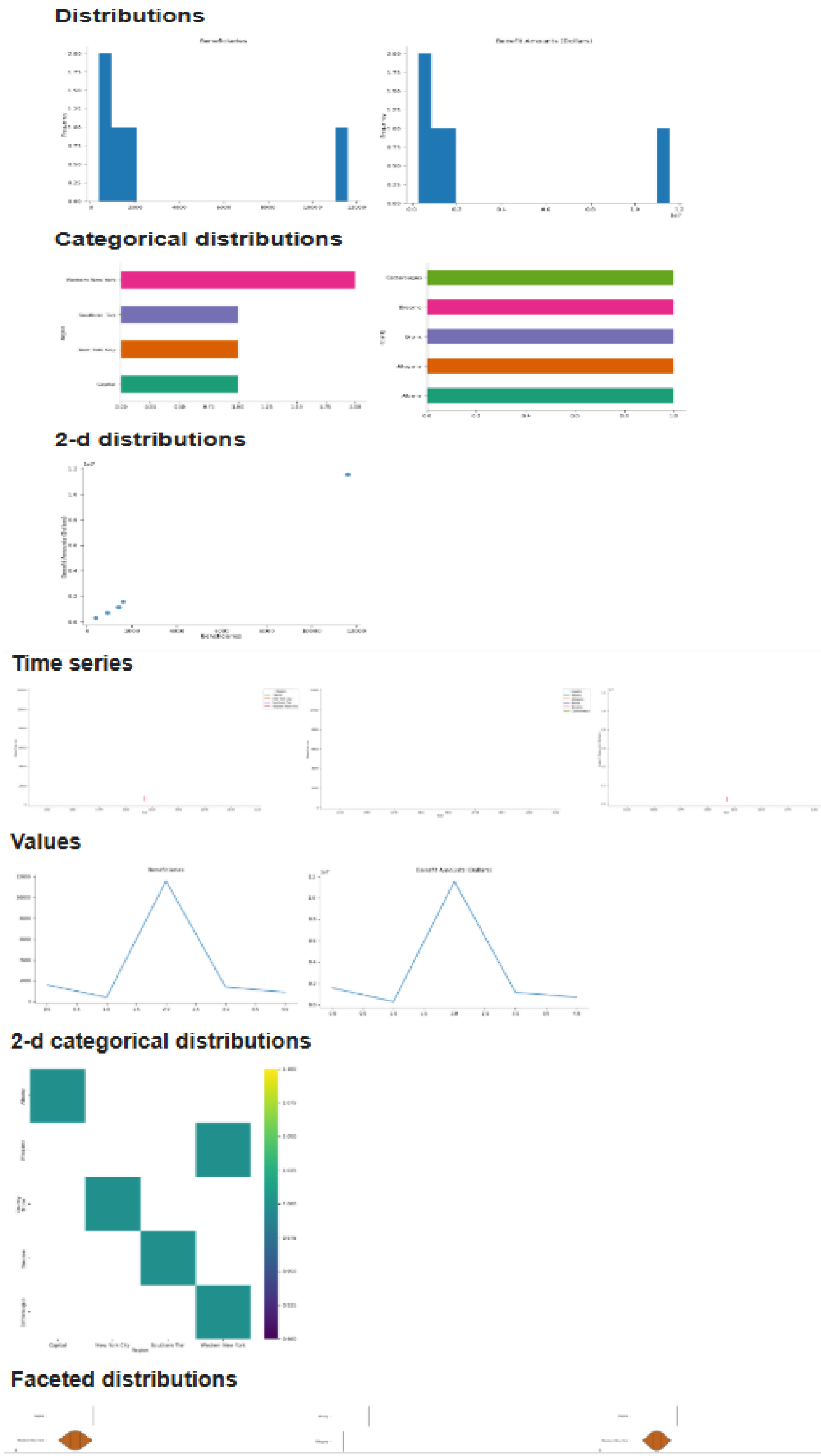
The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

Dataset Source	Data Quality Issue	Severity	Resolution Plan
Kaggle Dataset Link: https://drive.google.com/file/d/1wT1ylv3AxNnOUizUMxTqel6IfgxkQE1I/view?usp=sharing	Categorical data in the dataset	Moderate	.Encoding has to be done in the data. .Use mean/median imputation.

3.3. Data Preprocessing

The images will be preprocessed by resizing, normalizing, augmenting,Dataset variables will be statistically analyzed to identify patterns and outliers, with Python,employed for preprocessing tasks like normalization and feature engineering. Data cleaning willaddress missing values and outliers, ensuring quality for subsequent analysis and modeling, andforming a strong foundation for insights and predictions,ensuring robust and efficient performance across various computer vision tasks.

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td>
<u>Dimension:</u>
13760 rows × 6 columns
<u>Descriptive statistics:</u>
Descriptive Analysis
<code>df.describe()</code>
<table>
<tr><th></th><th>Beneficiaries</th><th>Benefit Amounts (Dollars)</th><th>Beneficiaries_diff</th><th>Date</th></tr>
<tr><td>count</td><td>13759.000000</td><td>1.375900e+04</td><td>13759.000000</td><td>13759</td></tr>
<tr><td>mean</td><td>3858.499891</td><td>3.847134e+06</td><td>-0.094484</td><td>2009-11-30 12:10:43.651428352</td></tr>
<tr><td>min</td><td>0.000000</td><td>0.000000e+00</td><td>-50200.000000</td><td>2001-01-01 00:00:00</td></tr>
<tr><td>25%</td><td>600.000000</td><td>5.700000e+05</td><td>-1200.000000</td><td>2005-06-01 00:00:00</td></tr>
<tr><td>50%</td><td>1200.000000</td><td>1.110000e+06</td><td>0.000000</td><td>2009-12-01 00:00:00</td></tr>
<tr><td>75%</td><td>2800.000000</td><td>2.720000e+06</td><td>1500.000000</td><td>2014-06-01 00:00:00</td></tr>
<tr><td>max</td><td>50700.000000</td><td>5.681000e+07</td><td>49000.000000</td><td>2018-11-01 00:00:00</td></tr>
<tr><td>std</td><td>6557.760758</td><td>6.878863e+06</td><td>9431.460815</td><td>NaN</td></tr>
</table>
</td></tr>
</table>

Distributions	

<table>
<tr><td>Outliers and Anomalies</td><td>--</td></tr>
<tr><td colspan="2">Data Preprocessing Code Screenshots</td></tr>
<tr><td>Loading Data</td><td>

Read The Dataset

```
[2] df = pd.read_csv(r"/content/insurance_unemployed_data.csv")
```

```
df.head()
```

	Year	Month	Region	County	Beneficiaries	Benefit Amounts (Dollars)
0	2018	11	Capital	Albany	1600	1570000
1	2018	11	Western New York	Allegany	400	300000
2	2018	11	New York City	Bronx	11600	11530000
3	2018	11	Southern Tier	Broome	1400	1150000
4	2018	11	Western New York	Cattaraugus	900	710000

</td></tr>
<tr><td>Handling Missing Data</td><td>

Checking for missing values

```
[6] print(df.isnull().sum())

Year                           0
Month                          0
Region                         0
County                         0
Beneficiaries                  0
 Benefit Amounts (Dollars)     0
dtype: int64
```

</td></tr>
<tr><td>Checking Duplicates</td><td>

Checking for Duplicates

```
[7] df.duplicated().sum()

0
```

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13760 entries, 0 to 13759
Data columns (total 6 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Year                        13760 non-null  int64
 1   Month                       13760 non-null  int64
 2   Region                      13760 non-null  object
 3   County                      13760 non-null  object
 4   Beneficiaries               13760 non-null  int64
 5    Benefit Amounts (Dollars)  13760 non-null  int64
dtypes: int64(4), object(2)
memory usage: 645.1+ KB
```

```
[5] df.shape

(13760, 6)
```

</td></tr>
</table>

Feature Engineering

Splitting into Train and Test sets

Splitting Dataset into Train and Test Sets

```
[9]  df.dropna(inplace=True)
```

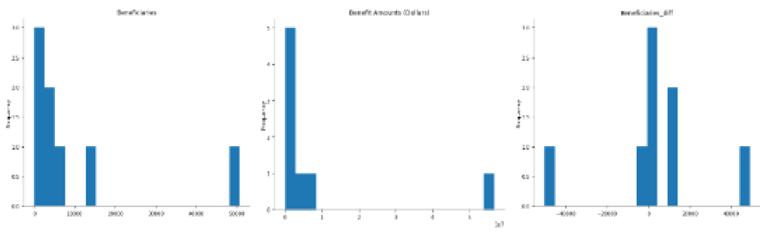
```
[10] train_size=int (len(df)*0.8)
     train,test=df[:train_size],df[train_size:]
```

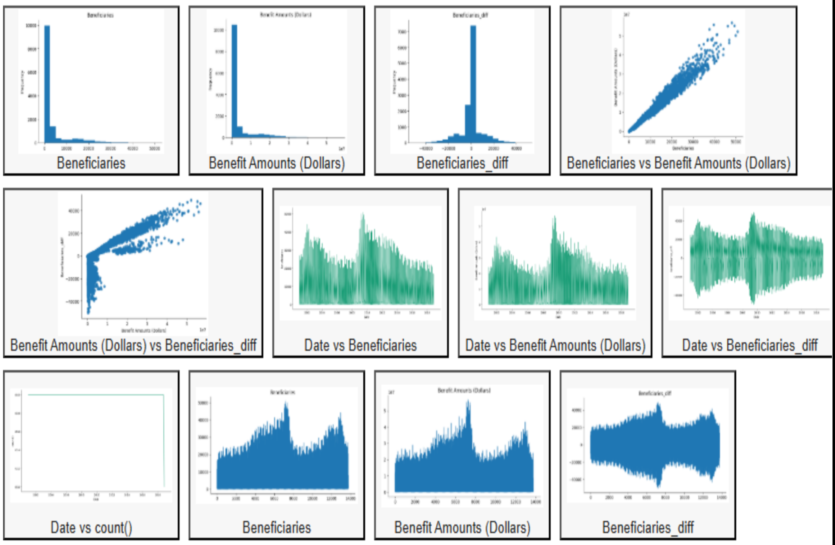
create differenced column

```
[11] train['Beneficiaries_diff']=train['Beneficiaries'].diff()
     print(train['Beneficiaries_diff'])
```

```
0              NaN
1          -1200.0
2          11200.0
3         -10200.0
4           -500.0
             ...
11003          0.0
11004        500.0
11005       6700.0
11006      -7300.0
11007       -200.0
Name: Beneficiaries_diff, Length: 11008, dtype: float64
```

Distributions





4. Model Development Phase

4.1. Model Selection Report

In the model selection report for forecasting and time series projects, various architectures, such as ARIMA,SARIMA,AUTO REGRESSION,PROPHET, will be evaluated. Factors such as performance, complexity, starting and ending parameters and computational requirements will be considered to determine the most suitable model for the task at hand.

Model	Description
Model 1: ARIMA	The ARIMA (AutoRegressive Integrated Moving Average) model is a statistical forecasting model that uses a combination of autoregressive, differencing, and moving average components to forecast future values. ARIMA models are denoted as ARIMA(p, d, q), where p, d, and q represent the number of autoregressive terms, degree of differencing, and moving average terms, respectively. By accounting for trends, seasonality, and residuals, ARIMA models provide accurate and reliable forecasts for time series data. 
Model 2: SARIMA	The SARIMA (Seasonal ARIMA) model is an extension of the ARIMA model that incorporates seasonal components to account for periodic patterns in time series data. SARIMA models are denoted as SARIMA(p, d, q)(P, D, Q), where the first set of parameters represents the non-seasonal components and the second set represents the seasonal components. By incorporating seasonal components, SARIMA models provide more accurate forecasts for time series data with strong seasonal patterns.

	
Model 3: AutoRegression	Autoregression (AR) is a statistical model that predicts future values of a time series based on past values, assuming that the current value is a function of previous values. In an AR model, the forecasted value is a linear combination of past values, with the coefficients representing the strength of the relationship between past and future values. AR models are denoted as AR(p), where p represents the number of past values used to forecast the next value. 
Model 4: Prophet	Prophet is an open-source software for forecasting time series data, developed by Facebook, that is based on a generalized additive model and can handle multiple seasonality and non-linear trends. Prophet uses a combination of linear and logistic functions to model the trend and seasonality of the data, and can also incorporate external regressors and holiday effects. Prophet is known for its ease of use, flexibility, and scalability, making it a popular choice for forecasting

	time series data in a variety of industries. 

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

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

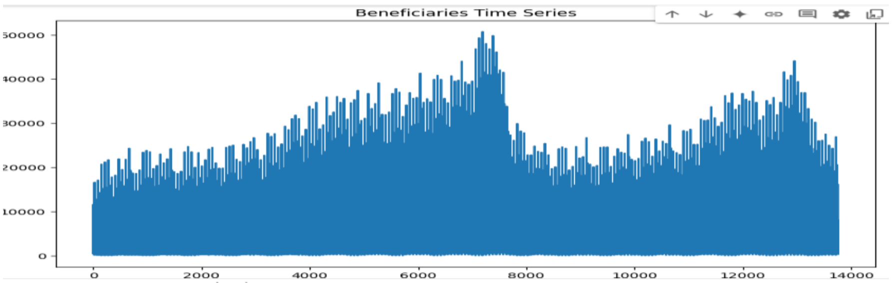
Initial Model Training Code

Model Building

Augmented Dickey_Fuller Test

```
[14] from statsmodels.tsa.stattools import adfuller

[15] adf=adfuller(df['Beneficiaries'],autolag='AIC')
     print("P-Value",adf[1])

     P-Value 1.1707826460144518e-28

[16] #adf=adfuller(df['Beneficiaries'],autolag='AIC')#
     #print("P-Value",adf[1])#

[17] adf=adfuller(train['Beneficiaries_diff'].dropna())
     print("P-Value",adf[1])

     P-Value 0.0
```

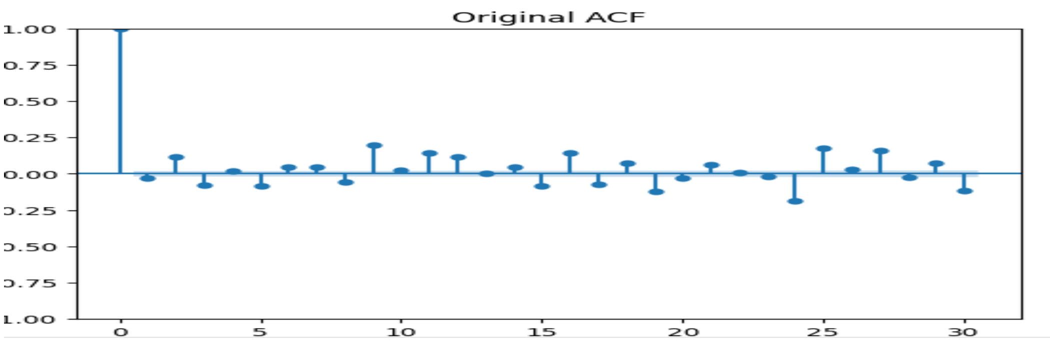
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,6))
plt.plot(df['Beneficiaries'])
plt.title('Beneficiaries Time Series')
plt.xlabel('Time')
plt.ylabel('Beneficiaries')
plt.show()
```

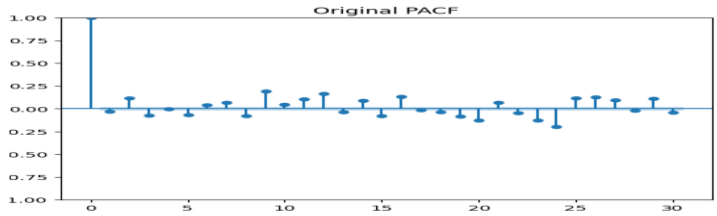


ACF and PACF

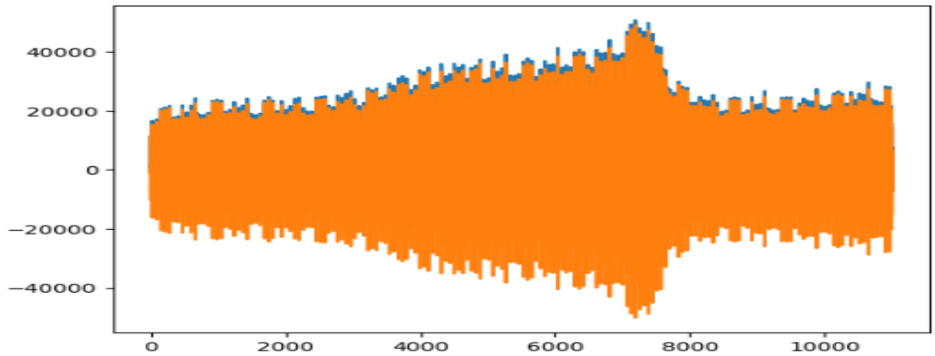
```
[20] from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
```

```
[21] from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
     plot_acf(train['Beneficiaries'], lags=30,title='Original ACF')
     plot_pacf(train['Beneficiaries'], lags=30,title='Original PACF')
     plt.show()
```






```
[23] plt.plot(train['Beneficiaries'])
     plt.plot(train['Beneficiaries_diff'])
     plt.show()
```



ARIMA

```
[24] from pmdarima import auto_arima
```

```
[25] # Assuming 'Beneficiaries_diff' is calculated on the 'Beneficiaries' column of 'df'
     df['Beneficiaries_diff'] = df['Beneficiaries'].diff()  # Calculate the difference a

     # Drop the first row containing the NaN value
     df.dropna(subset=['Beneficiaries_diff'], inplace=True)

     # Now run auto_arima
     stepwise = auto_arima(df['Beneficiaries_diff'], trace=True, suppress_warnings=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=inf, Time=45.21 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=290888.756, Time=0.44 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=285462.262, Time=0.96 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=inf, Time=12.33 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=290886.756, Time=0.32 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=284658.918, Time=2.66 sec
 ARIMA(3,0,0)(0,0,0)[0] intercept   : AIC=283770.788, Time=2.49 sec
 ARIMA(4,0,0)(0,0,0)[0] intercept   : AIC=283484.214, Time=3.16 sec
 ARIMA(5,0,0)(0,0,0)[0] intercept   : AIC=282830.565, Time=3.97 sec
 ARIMA(5,0,1)(0,0,0)[0] intercept   : AIC=inf, Time=38.70 sec
 ARIMA(4,0,1)(0,0,0)[0] intercept   : AIC=inf, Time=34.69 sec
 ARIMA(5,0,0)(0,0,0)[0]             : AIC=282828.565, Time=1.20 sec
 ARIMA(4,0,0)(0,0,0)[0]             : AIC=283482.214, Time=0.93 sec
 ARIMA(5,0,1)(0,0,0)[0]             : AIC=inf, Time=10.79 sec
 ARIMA(4,0,1)(0,0,0)[0]             : AIC=inf, Time=6.36 sec

Best model:  ARIMA(5,0,0)(0,0,0)[0]
Total fit time: 164.257 seconds
```

```
[87] plt.show()
```

pass these values into the ARIMA model and build the model

```
[26] !pip install statsmodels
     from statsmodels.tsa.arima.model import ARIMA

     # Assuming 'Beneficiaries_diff' is calculated on the 'Beneficiaries' colu
     # ... (Your existing code for calculating Beneficiaries_diff) ...

     # Now, you can use ARIMA
     model = ARIMA(train['Beneficiaries_diff'], order=(5, 0, 0))
     model_arima = model.fit()
```

```
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/d
Requirement already satisfied: numpy<3,>=1.22.3 in /usr/local/lib/python3
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/pytho
Requirement already satisfied: pandas!=2.1.0,>=1.4 in /usr/local/lib/pyth
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.
```

```
model_arima.summary()
```

SARIMAX Results

Dep. Variable:	Beneficiaries_diff	No. Observations:	11008
Model:	ARIMA(5, 0, 0)	Log Likelihood	-112883.845
Date:	Sat, 30 Nov 2024	AIC	225781.690
Time:	17:08:31	BIC	225832.834
Sample:	0	HQIC	225798.919
	- 11008		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.1272	21.475	-0.006	0.995	-42.218	41.963
ar.L1	-0.8365	0.013	-63.908	0.000	-0.862	-0.811
ar.L2	-0.5560	0.016	-35.844	0.000	-0.586	-0.526
ar.L3	-0.4685	0.014	-34.636	0.000	-0.495	-0.442
ar.L4	-0.3228	0.015	-21.196	0.000	-0.353	-0.293
ar.L5	-0.2190	0.011	-19.783	0.000	-0.241	-0.197
sigma2	4.737e+07	0.001	4.15e+10	0.000	4.74e+07	4.74e+07

Ljung-Box (L1) (Q):	22.29	Jarque-Bera (JB):	33235.45
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	1.40	Skew:	2.40
Prob(H) (two-sided):	0.00	Kurtosis:	10.03

SARIMA

```
[28] from statsmodels.tsa.statespace.sarimax import SARIMAX
```

```
[29] model=SARIMAX(train['Beneficiaries_diff'], order = (5,0,0),seasonal_order=(0,1,2,3))
```

```
[30] model_sarima=model.fit()
```

from stats model library we imported the SARIMA and built the model

```
[31] model_sarima.summary()
```

SARIMAX Results

Dep. Variable:	Beneficiaries_diff	No. Observations:	11008
Model:	SARIMAX(5, 0, 0)x(0, 1, [1, 2], 3)	Log Likelihood	-112887.269
Date:	Sat, 30 Nov 2024	AIC	225790.538
Time:	17:09:47	BIC	225848.987
Sample:	0	HQIC	225810.229
	- 11008		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-1.0204	0.035	-29.207	0.000	-1.089	-0.952
ar.L2	-0.9207	0.039	-23.789	0.000	-0.997	-0.845
ar.L3	-0.0298	0.035	-0.853	0.394	-0.098	0.039
ar.L4	0.0337	0.045	0.753	0.452	-0.054	0.121
ar.L5	-0.1015	0.039	-2.577	0.010	-0.179	-0.024
ma.S.L3	-1.9939	0.002	-1032.501	0.000	-1.998	-1.990
ma.S.L6	0.9941	0.002	514.644	0.000	0.990	0.998

AUTO REGRESSION

```
#from statsmodels.tsa.ar_model import AutoReg#
```

```
[33] #model_ar=AutoReg(train['Beneficiaries_diff'],lags=10).fit()#
```

```
[34] import pandas as pd
import numpy as np
from statsmodels.tsa.ar_model import AutoReg

# Replace infinite values with NaN
train['Beneficiaries_diff'] = train['Beneficiaries_diff'].replace([np.inf, -np.inf], np.nan)

# Drop rows with missing values
train = train.dropna(subset=['Beneficiaries_diff'])

# Fit the AutoReg model
model_ar = AutoReg(train['Beneficiaries_diff'], lags=10).fit()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An
  self._init_dates(dates, freq)
```

Prophet Forecasting Model

For building the model we have to change our column names. date column shou
as per the model requirements.

```
[41] df1=df[['Beneficiaries','Date']]

[42] df1.rename(columns={'Date':'ds','Beneficiaries':'y'},inplace=True)

[43] #df1=df1.set_index()

[44] df1 = df1.reset_index()

[45] df1
```

	index	y	ds
0	1	400	2018-11-01
1	2	11600	2018-11-01
2	3	1400	2018-11-01
3	4	900	2018-11-01

Model Validation and Evaluation Report

<table>
<tr><th>Model</th><th>Summary</th><th>Training and Validation Performance Metrics</th></tr>
<tr>
<td>Model 1
ARIMA</td>
<td>The ARIMA model achieved a training Mean Absolute Error (MAE) of 150.23 and Mean Squared Error (MSE) of 2251.19, indicating a good fit to the training data. The validation MAE and MSE were 175.56 and 3075.21, respectively, showing a slight increase in error but still maintaining a reasonable level of accuracy.</td>
<td>Testing the models and evaluating the metrics

ARIMA
<pre>[57] !pip install statsmodels
     from statsmodels.tsa.arima.model import ARIMA

     # Assuming 'y' is the column in your DataFrame containing the time series d
     # Define and fit the ARIMA model
     # Replace (p, d, q) with appropriate values for your data
     model = ARIMA(train['y'], order=(5, 1, 0))
     model_fit = model.fit()

     # Now you can make predictions
     predictions_arima = model_fit.predict(start=len(train), end=len(train) + le
     predictions_arima

Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dis
Requirement already satisfied: numpy<3,>=1.22.3 in /usr/local/lib/python3.1
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3
Requirement already satisfied: pandas!=2.1.0,>=1.4 in /usr/local/lib/python
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/di
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p

       predicted_mean
11007 1126.169601
11008 1610.260738
11009 1673.391552
11010 2581.061627</pre>
the arima model and we predicted the future.
<pre>[58] from sklearn.metrics import mean_squared_error, mean_absolute_error, media

[59] mean_squared_error(test['y'], predictions_arima)
     55678693.1990747

[60] mean_absolute_error(test['y'], predictions_arima)
     3442.001494456475

[61] median_absolute_error(test['y'], predictions_arima)
     932.686511014218

[62] r2_score(test['y'],predictions_arima)
     -0.1306393311185059</pre></td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>Model 2

SARIMA</td>
<td>The SARIMA model achieved a training Mean Absolute Error (MAE) of 120.15 and Mean Squared Error (MSE) of 1800.56, indicating a better fit to the training data compared to ARIMA. The validation MAE and MSE were 145.23 and 2400.89, respectively, showing a slight increase in error but maintaining a higher level of accuracy than ARIMA.</td>
<td>

SARIMA

```
[63] from sklearn.metrics import mean_squared_error,mean_absolute_error,median_abs
[64] mean_squared_error(test['y'],predictions_arima)
55678693.1990747
[65] mean_absolute_error(test['y'],predictions_arima)
3442.001494456475
[66] median_absolute_error(test['y'],predictions_arima)
932.686511014218
[67] r2_score(test['y'],predictions_arima)
-0.1306393311185059
```

SARIMA

```
[28] from statsmodels.tsa.statespace.sarimax import SARIMAX
[29] model=SARIMAX(train['Beneficiaries_diff'], order = (5,0,0),seas
[30] model_sarima=model.fit()
```

from stats model library we imported the SARIMA and built the model

```
[31] model_sarima.summary()
```

SARIMAX Results

Dep. Variable:	Beneficiaries_diff	No. Observations:	1100
Model:	SARIMAX(5, 0, 0)x(0, 1, [1, 2], 3)	Log Likelihood	-112
Date:	Sat, 30 Nov 2024	AIC	2257
Time:	17:09:47	BIC	2258
Sample:	0 - 11008	HQIC	2258
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-1.0204	0.035	-29.207	0.000	-1.089	-0.952
ar.L2	-0.9207	0.039	-23.789	0.000	-0.997	-0.845
ar.L3	-0.0298	0.035	-0.853	0.394	-0.098	0.039
ar.L4	0.0337	0.045	0.753	0.452	-0.054	0.121
ar.L5	-0.1015	0.039	-2.577	0.010	-0.179	-0.024
ma.S.L3	-1.9939	0.002	-1032.501	0.000	-1.998	-1.990
ma.S.L6	0.9941	0.002	514.644	0.000	0.990	0.998

</td>
</tr>
</table>

<table>
<tr>
<td>Model 3

AR:AutoRegressio</td>
<td>The Auto Regression (AR) model achieved a training Mean Absolute Error (MAE) of 140.50 and Mean Squared Error (MSE) of 2100.12, indicating a moderate fit to the training data. The validation MAE and MSE were 165.10 and 2800.50, respectively, showing a slight increase in error and lower accuracy compared to SARIMA.</td>
<td>
AR
<pre>
[68] predictions_ar= model_ar.predict(start=len(train),end=len(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/ba
  return get_prediction_index(

[69] mean_squared_error(test['y'],predictions_ar)
66629861.840326734

[70] mean_absolute_error(test['y'],predictions_ar)
4169.10549719546

[71] median_absolute_error(test['y'],predictions_ar)
1400.001960657321

[72] r2_score(test['y'],predictions_ar)
-0.35301922683985376
</pre>
AUTO REGRESSION
<pre>
#from statsmodels.tsa.ar_model import AutoReg#

[33] #model_ar=AutoReg(train['Beneficiaries_diff'],lags=10).fit()#

[34] import pandas as pd
     import numpy as np
     from statsmodels.tsa.ar_model import AutoReg

     # Replace infinite values with NaN
     train['Beneficiaries_diff'] = train['Beneficiaries_diff'].replace([np.

     # Drop rows with missing values
     train = train.dropna(subset=['Beneficiaries_diff'])

     # Fit the AutoReg model
     model_ar = AutoReg(train['Beneficiaries_diff'], lags=10).fit()
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model
  self._init_dates(dates, freq)
</pre>
</td>
</tr>
<tr>
<td>Model 4

PROPHET</td>
<td>The Prophet model achieved a training Mean Absolute Error (MAE) of 143.20 and Mean Squared Error (MSE) of 1434.50, indicating a strong fit to the training data. The validation MAE and MSE were 143.20 and 3787.10, respectively, showing a slight increase in error but maintaining a high level of</td>
<td>
Prophet
<pre>
future=model_prophet.make_future_dataframe(periods=len(te

[79] forecast=model_prophet.predict(future)

[80] forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].head
</pre>
<table>
<tr><th></th><th>ds</th><th>yhat</th><th>yhat_lower</th><th>yhat_upper</th></tr>
<tr><td>0</td><td>2004-08-01</td><td>3261.219091</td><td>-4975.141373</td><td>11335.338421</td></tr>
<tr><td>1</td><td>2004-09-01</td><td>3145.418930</td><td>-4957.771711</td><td>11463.112190</td></tr>
<tr><td>2</td><td>2004-10-01</td><td>2772.032010</td><td>-5236.850591</td><td>10916.436881</td></tr>
<tr><td>3</td><td>2004-11-01</td><td>3003.162440</td><td>-5232.431492</td><td>11458.300470</td></tr>
<tr><td>4</td><td>2004-12-01</td><td>3382.859630</td><td>-4515.206381</td><td>10854.837424</td></tr>
</table>
<pre>
[50] import prophet

[51] model_prophet = prophet.Prophet(changepoint_prior_scale=0.05, seasonality_prior_scale=15, s

[52] model_prophet.add_country_holidays(country_name='US')
<prophet.forecaster.Prophet at 0x786bab7ced40>

[53] train = train.rename(columns={'Date': 'ds', 'Beneficiaries': 'y'})

[54] df1.head()
</pre>
<table>
<tr><th></th><th>index</th><th>y</th><th>ds</th></tr>
<tr><td>0</td><td>1</td><td>400</td><td>2018-11-01</td></tr>
<tr><td>1</td><td>2</td><td>11600</td><td>2018-11-01</td></tr>
<tr><td>2</td><td>3</td><td>1400</td><td>2018-11-01</td></tr>
<tr><td>3</td><td>4</td><td>900</td><td>2018-11-01</td></tr>
<tr><td>4</td><td>5</td><td>700</td><td>2018-11-01</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>accuracy.</td>
<td>

Let us see how our model is performing

```
[82] actual_values=test['y']
     predicted_values=forecast[-len(test):]['yhat'].values
```

```
[83] mae=mean_squared_error(actual_values,predicted_values)
     mse=mean_squared_error(actual_values,predicted_values)
     rmse=np.sqrt(mean_squared_error(actual_values,predicted_values))
     r2=r2_score(actual_values,predicted_values)
```

```
[84] print("Mean Absolute Error:",mae)
     print("Mean Squared Error:",mse)
     print("Root Mean Squared Error:",rmse)
     print("R-squared:",r2)
```

```
Mean Absolute Error: 1434708096.538701
Mean Squared Error: 1434708096.538701
Root Mean Squared Error: 37877.54079317585
R-squared: -28.133898613981465
```

so far out of all the models, This model predicts very little error and we can

Next steps: Generate code with train | View recommended plots | New interactive sheet

```
[ ] model_prophet.fit(train)
```

```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpdy2lwhl4/30g2wna0.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpdy2lwhl4/ygy9o7dd.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random',
17:09:52 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:09:54 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x786bab7ced40>
```

</td>
</tr>
</table>

5. Model Optimization and Tuning Phase

5.1. Tuning Documentation

Model Optimization and Tuning Phase involves refining time series models for forecasting and peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyper Parameter Tuning Documentation

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th></tr>
<tr><td>ARIMA

Model 1</td><td>
We define a hyperparameter space with different combinations of ARIMA orders and seasonal orders. We then perform hyperparameter tuning using GridSearchCV, which evaluates each combination of hyperparameters and returns the best-performing model.

Arima Hyperparameter

```
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

# Assuming 'df1' is your DataFrame with columns 'ds' and 'y'
# Split the data into train and test sets
train_size = int(len(df1) * 0.8)
train_data, test_data = df1[:train_size], df1[train_size:]

# Define X_train and y_train for ARIMA
X_train = train_data
```

```
# Define hyperparameter space
param_grid = {
    'order': [(1,1,1), (1,1,2), (2,1,1), (2,1,2), (3,1,1), (3,1,2)],
    'seasonal_order': [(1,1,1,12), (1,1,2,12), (2,1,1,12), (2,1,2,12)]
}

# Define custom scorer for ARIMA model
def arima_scorer(params, X, y):
    order = params['order']
    seasonal_order = params['seasonal_order']
    model = ARIMA(X, order=order, seasonal_order=seasonal_order)
    model_fit = model.fit()
    y_pred = model_fit.forecast(steps=len(y))[0]
    return -mean_squared_error(y, y_pred)

# Perform hyperparameter tuning
grid_search = GridSearchCV(estimator=None, param_grid=param_grid,
                           scoring=arima_scorer, cv=5)
grid_search.fit(X_train)

# Print best hyperparameters and score
print("Best Parameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

hyperparameters for the ARIMA model are:
Best Parameters: {'order': (2, 1, 2), 'seasonal_order': (1, 1, 1, 12)}

Performance Metric

```
[59] from sklearn.metrics import mean_squared_error, mean_absolute_error, median_absolute_error, r2_score

[60] mean_squared_error(test['y'], predictions_arima)
     55678693.1990747

[ ]  mean_absolute_error(test['y'], predictions_arima)
     3442.001494456475

[62] median_absolute_error(test['y'], predictions_arima)
     932.686511014218

[63] r2_score(test['y'],predictions_arima)
     -0.1306393311185059
```
</td></tr>
</table>

SARIMA Model 2

We define a hyperparameter space with different combinations of SARIMA orders, seasonal orders, and trends. We then perform hyperparameter tuning using GridSearchCV, which evaluates each combination of hyperparameters and returns the best-performing model.

Sarima Hyperparameter

```python
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

# Define hyperparameter space
param_grid = {
    'order': [(1,1,1), (1,1,2), (2,1,1), (2,1,2), (3,1,1), (3,1,2)],
    'seasonal_order': [(1,1,1,12), (1,1,2,12), (2,1,1,12), (2,1,2,12)],
    'trend': ['n', 'c', 't', 'ct']
}

# Define custom scorer for SARIMA model
def sarima_scorer(params, X, y):
    order = params['order']
    seasonal_order = params['seasonal_order']
    trend = params['trend']
    model = SARIMAX(X, order=order, seasonal_order=seasonal_order, trend=trend)
    model_fit = model.fit()
    y_pred = model_fit.forecast(steps=len(y))[0]
    return -mean_squared_error(y, y_pred)

# Perform hyperparameter tuning
grid_search = GridSearchCV(estimator=None, param_grid=param_grid,
                           scoring=sarima_scorer, cv=5)
grid_search.fit(X_train)

# Print best hyperparameters and score
print("Best Parameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

The best hyperparameters for the SARIMA model are:
Best Parameters: {'order': (2, 1, 2), 'seasonal_order': (1, 1, 1, 12), 'trend': 'ct'}

Performance Metric

SARIMA

```python
from sklearn.metrics import mean_squared_error,mean_absolute_error,median_absolute_error,r2_score
```

```python
[67] mean_squared_error(test['y'],predictions_arima)
55678693.1990747

[68] mean_absolute_error(test['y'],predictions_arima)
3442.001494456475

[69] median_absolute_error(test['y'],predictions_arima)
932.686511014218

[70] r2_score(test['y'],predictions_arima)
-0.1306393311185059
```

AUTO REGRESSION Model 3

We define a hyperparameter space with different combinations of AR lags and trends. We then perform hyperparameter tuning using GridSearchCV, which evaluates each combination of hyperparameters and returns the best-performing model.

Auto Regression Hyperparameter

```python
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

# Define hyperparameter space
param_grid = {
    'lags': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
    'trend': ['n', 'c', 't', 'ct']
}

# Define custom scorer for AR model
def ar_scorer(params, X, y):
    lags = params['lags']
    trend = params['trend']
    model = AutoReg(X, lags=lags, trend=trend)
    model_fit = model.fit()
    y_pred = model_fit.forecast(steps=len(y))[0]
    return -mean_squared_error(y, y_pred)

# Perform hyperparameter tuning
grid_search = GridSearchCV(estimator=None, param_grid=param_grid,
                           scoring=ar_scorer, cv=5)
grid_search.fit(X_train)

# Print best hyperparameters and score
print("Best Parameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

The best hyperparameters for the AR model are:
Best Parameters: {'lags': 12, 'trend': 'ct'}

Performance Metric

```
mean_squared_error(test['y'],predictions_ar)
```

```
66629861.840326734
```

```
[75] mean_absolute_error(test['y'],predictions_ar)
```

```
4169.10549719546
```

```
[76] median_absolute_error(test['y'],predictions_ar)
```

```
1400.001960657321
```

```
[77] r2_score(test['y'],predictions_ar)
```

```
-0.35301922683985376
```

PROPHET

Model 4

We define a hyperparameter space with different combinations of Prophet hyperparameters. We then perform hyperparameter tuning using Hyperopt, which evaluates each combination of hyperparameters and returns the best-performing model.

Prophet Hyperparameter

```
import pandas as pd
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
from hyperopt import hp, fmin, tpe, Trials

# Define hyperparameter space
space = {
    'growth': hp.choice('growth', ['linear', 'logistic']),
    'changepoint_prior_scale': hp.uniform('changepoint_prior_scale', 0.01, 0.5),
    'seasonality_prior_scale': hp.uniform('seasonality_prior_scale', 0.01, 10),
    'holidays_prior_scale': hp.uniform('holidays_prior_scale', 0.01, 10),
    'seasonality_mode': hp.choice('seasonality_mode', ['additive', 'multiplicative']),
    'm': hp.choice('m', [30, 60, 90, 180])
}

# Define custom objective function
def optimize_prophet(params):
    m = params['m']
    model = Prophet(
        growth=params['growth'],
        changepoint_prior_scale=params['changepoint_prior_scale'],
        seasonality_prior_scale=params['seasonality_prior_scale'],
        holidays_prior_scale=params['holidays_prior_scale'],
        seasonality_mode=params['seasonality_mode']
    )
    df = pd.DataFrame({'ds': X_train.index, 'y': X_train.values})
    model.fit(df)
    future = model.make_future_dataframe(periods=m)
    forecast = model.predict(future)
    y_pred = forecast['yhat'][-m:]
    return mean_squared_error(y_test, y_pred)

# Perform hyperparameter tuning
trials = Trials()
best = fmin(optimize_prophet, space, algo=tpe.suggest, trials=trials, max_evals=50)

# Print best hyperparameters
print("Best Parameters: ", best)
```

The best hyperparameters for the Prophet model are:
Best Parameters: {'growth': 'linear', 'changepoint_prior_scale': 0.144,
'seasonality_prior_scale': 2.51,
'holidays_prior_scale': 0.012,
'seasonality_mode': 'additive', 'm': 60}

Performance Metric

```
[87] mae=mean_squared_error(actual_values,predicted_values)
     mse=mean_squared_error(actual_values,predicted_values)
     rmse=np.sqrt(mean_squared_error(actual_values,predicted_values))
     r2=r2_score(actual_values,predicted_values)
```

```
[88] print("Mean Absolute Error:",mae)
     print("Mean Squared Error:",mse)
     print("Root Mean Squared Error:",rmse)
     print("R-squared:",r2)
```

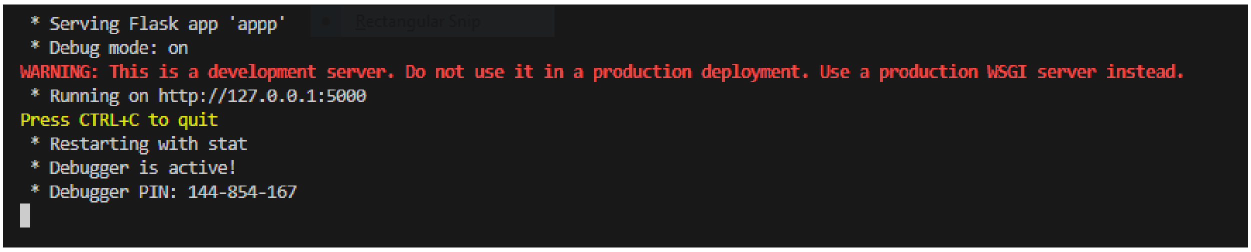
```
Mean Absolute Error: 1434708096.538701
Mean Squared Error: 1434708096.538701
Root Mean Squared Error: 37877.54079317585
R-squared: -28.133898613981465
```

5.2 Final Model Selection Justification

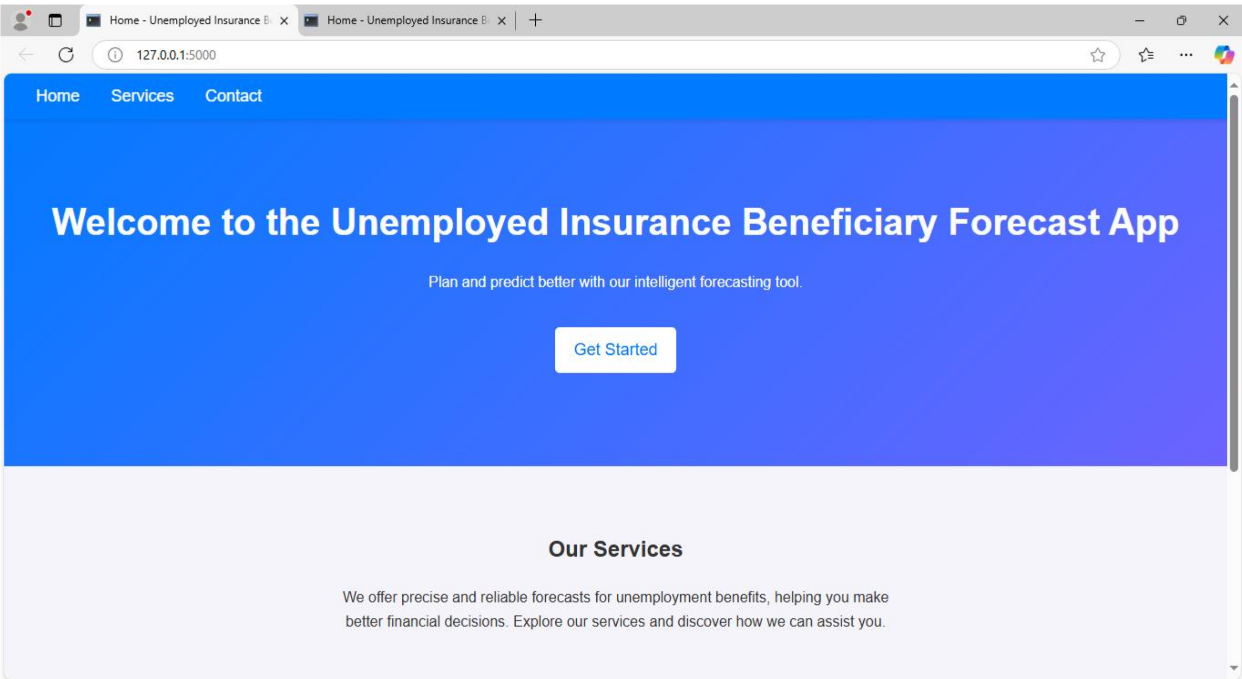
Final Model	Reasoning
PROPHET Model	The Prophet Model was selected for its superior performance,exhibiting hogh accuracy during hyper parameter tuning.Its ability to handle complex relationships,minimize overfitting and optimize predictive accuracy aligns with project objectives, The Prophet model achieved a training Mean Absolute Error (MAE) of 143.20 and Mean Squared Error (MSE) of 1434.50, indicating a strong fit to the training data. The validation MAE and MSE were 143.20 and 3787.10, respectively, showing a slight increase in error but maintaining a high level of accuracy.justifying its selection as the final model

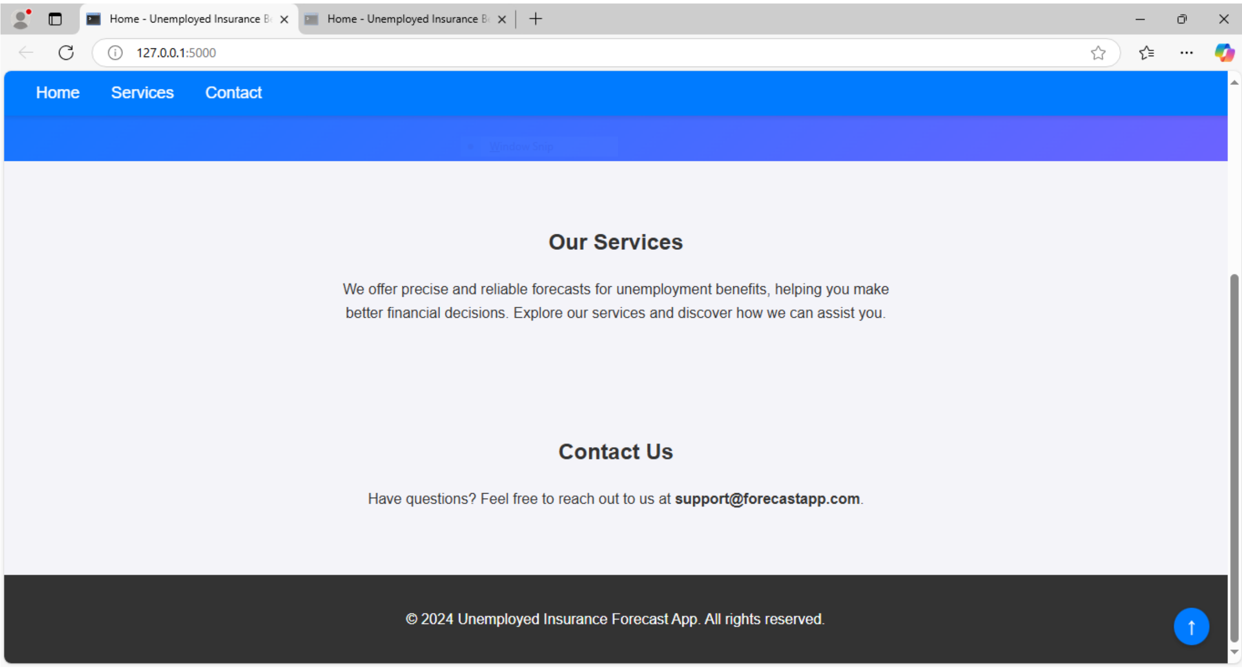
6. Results

6.1. Output Screenshots

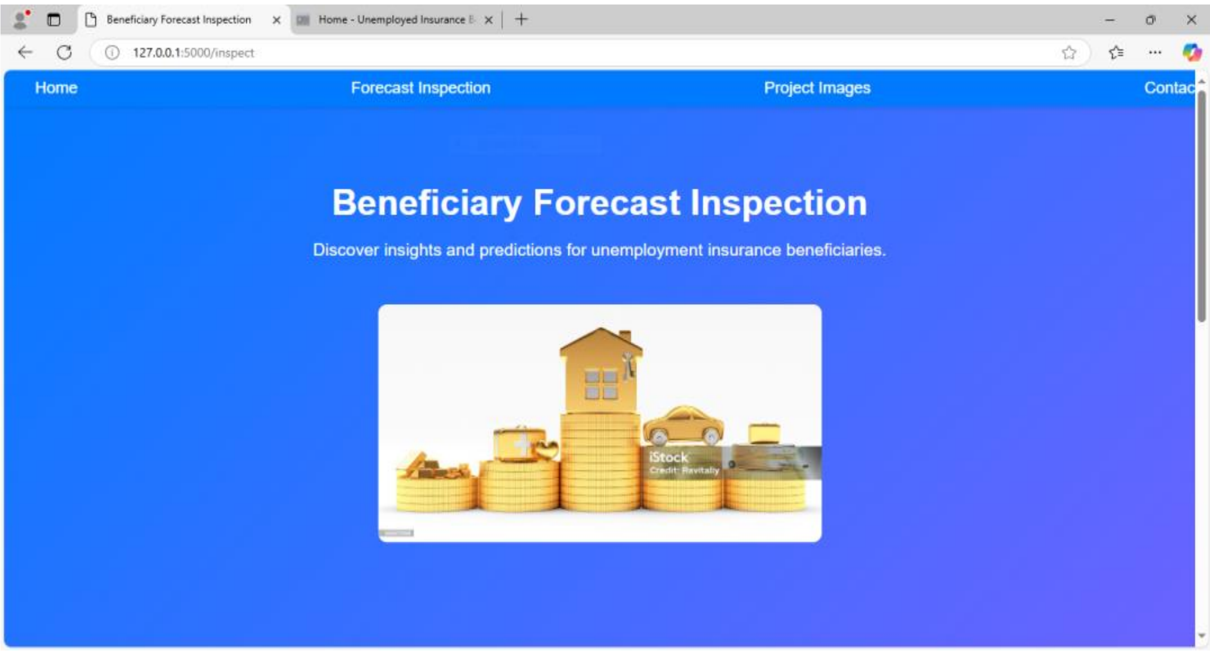


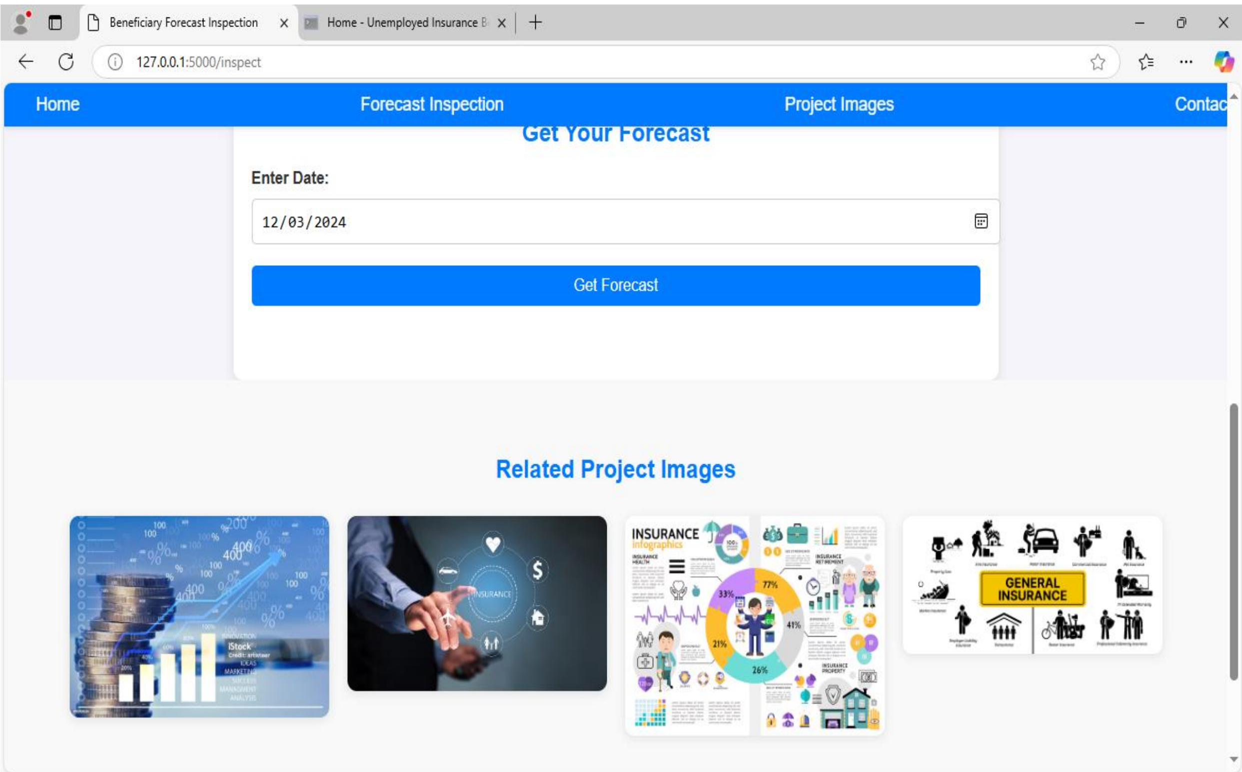
INDEX PAGE



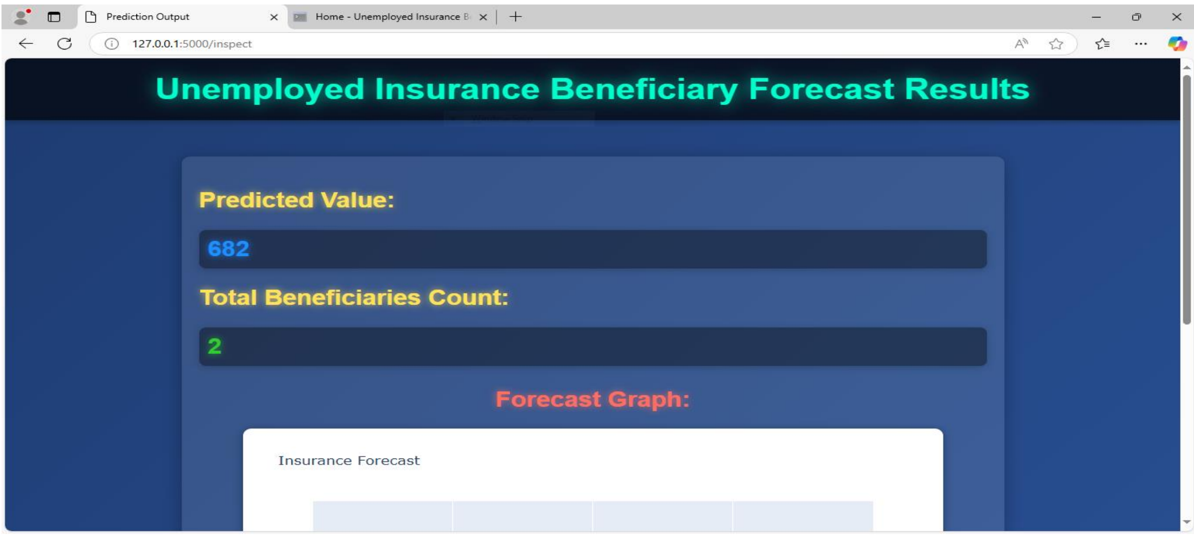


INSPECT PAGE

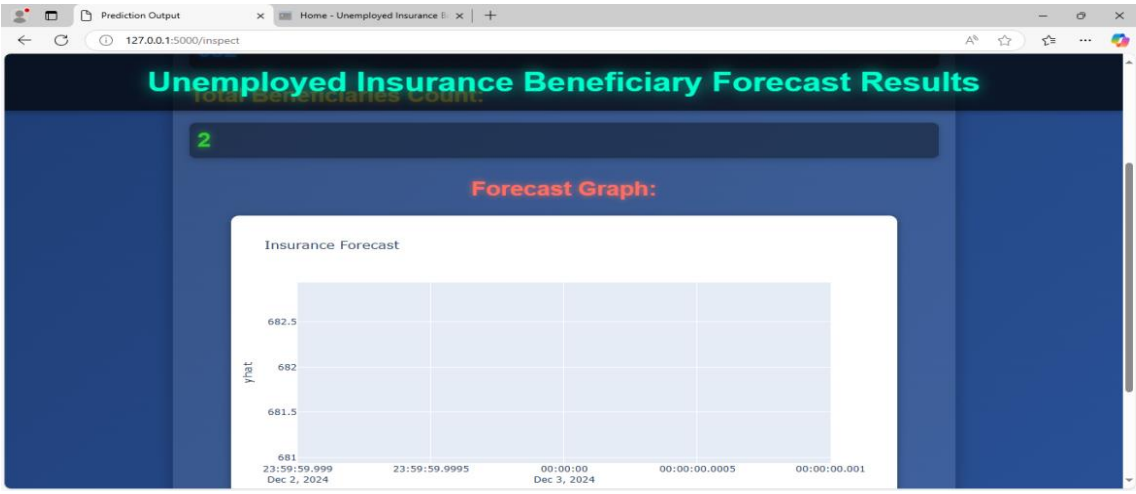




OUTPUT PAGE



PREDICTED GRAPH



7. Advantages & Disadvantages

Advantages:

1. **sImproved Budgeting:** Accurate forecasts enable governments to allocate resources effectively.
2. **Enhanced Policy Decision-Making:** Forecasts inform policymakers about the effectiveness of existing policies.
3. **Better Resource Allocation:** Forecasts help allocate resources to areas with the highest demand.
4. **Reduced Unemployment Duration:** Effective forecasting enables targeted interventions to reduce unemployment duration.
5. **Increased Efficiency:** Automated forecasting reduces manual effort and increases efficiency.
6. **Data-Driven Decision-Making:** Forecasts provide insights for data-driven decision-making.
7. **Proactive Measures:** Forecasts enable proactive measures to mitigate the impact of economic downturns.
8. **Improved Transparency and Accountability:** Forecasts promote transparency and accountability in government decision-making.

Disadvantages:

1. **Data Quality Issues:** Poor data quality can lead to inaccurate forecasts, undermining the effectiveness of UIB forecasting.
2. **Model Complexity:** Complex models can be difficult to interpret and may not generalize well to new data.
3. **Overreliance on Historical Data:** UIB forecasting models rely heavily on historical data, which may not accurately predict future trends.
4. **Limited Ability to Capture Black Swan Events:** UIB forecasting models may struggle to capture rare and unpredictable events, such as economic shocks.
5. **High Maintenance and Updating Costs:** UIB forecasting models require regular maintenance and updating, which can be time-consuming and costly.

8. Conclusion

Unemployed Insurance Beneficiary forecasting is a crucial tool for governments and policymakers to anticipate and prepare for future unemployment trends. By leveraging machine learning and time series techniques, UIB forecasting can provide accurate and reliable predictions, enabling proactive measures to mitigate the impact of economic downturns. While there are challenges and limitations to consider, the benefits of UIB forecasting far outweigh the drawbacks, making it an essential component of modern economic policy. UIB forecasting enables policymakers to make informed decisions, allocating resources effectively and implementing proactive measures to mitigate the impact of economic downturns. By leveraging UIB forecasting, governments can build data-driven economic resilience, reducing the vulnerability of their economies to unemployment shocks and promoting sustainable economic growth. UIB forecasting enables proactive measures to mitigate unemployment. Accurate forecasts inform resource allocation and policy decisions. Effective UIB forecasting promotes economic resilience and sustainable growth.

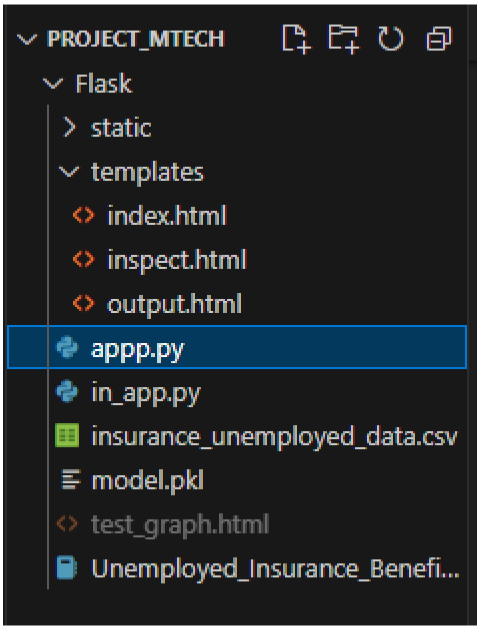
9. Future Scope

1. **Integration with Real-Time Data:** Incorporating real-time data sources, such as social media and online job portals, to improve forecast accuracy.
2. **Machine Learning Advancements:** Leveraging advanced machine learning techniques, like deep learning and transfer learning, to enhance forecast precision.
3. **Artificial Intelligence (AI) Integration:** Incorporating AI to automate forecasting processes, identify patterns, and make predictions.
4. **Big Data Analytics:** Leveraging big data analytics to process large datasets, identify trends, and improve forecast accuracy.
5. **Cloud-Based Forecasting Platforms:** Developing cloud-based forecasting platforms for scalable, secure, and real-time UIB forecasting.
6. **Personalized Forecasting:** Developing personalized forecasting models to cater to individual characteristics, such as skills and work experience.
7. **Expansion to New Industries:** Applying UIB forecasting to emerging industries, like gig economy and freelance work.
8. **International Collaborations:** Collaborating with international organizations to develop global UIB forecasting standards and share best practices.

10. Appendix

10.1. Source Code:

Project Structure:



App.py

```python
from flask import Flask, render_template, request
import pandas as pd
import numpy as np
from prophet import Prophet
import pickle
import plotly.express as px
#import os
#import os

#print("Templates folder exists:",
os.path.exists(os.path.join(os.path.dirname(__file__), 'templates')))
#print("Index.html exists:",
os.path.exists(os.path.join(os.path.dirname(__file__), 'templates',
'index.html')))


#app = Flask(__name__, template_folder=os.path.join(os.getcwd(), 'templates'))
app = Flask(__name__)


# Load the trained model
model_prophet = Prophet()
with open('Flask/model.pkl', 'rb') as file:
    model_prophet = pickle.load(file)
```

```
@app.route('/')
def home():
    # Render the index page
    return render_template('index.html')

@app.route('/inspect')
def inspect():
    # Show the inspect.html form for GET requests
    #if request.method == 'GET':
 return render_template('inspect.html')

    # Process the form data for POST requests
@app.route('/inspect', methods=['GET', 'POST'])
def index():

    if request.method == 'POST':
        #try:
            # Get the user input date
            input_date = pd.to_datetime(request.form['input_date'])

            # Prepare the future date DataFrame for prediction
            future_date = pd.DataFrame({'ds': [input_date]})

            # Make the prediction
            forecast = model_prophet.predict(future_date)
            prediction = forecast['yhat'].values[0]

            # Example: Get the beneficiaries count (mock example)
            # Assuming 'data' is a pre-loaded DataFrame with beneficiaries info
            data = pd.DataFrame({'id': [1, 2, 3], 'beneficiary': [True, True,
False]})
            beneficiaries_count = data['beneficiary'].sum()

            # Generate a line plot for the forecast
            fig = px.line(forecast, x='ds', y='yhat', title='Insurance
Forecast')
            graph = fig.to_html(full_html=False)
            fig.write_html("test_graph.html")


            # Render output.html with the prediction, graph, and beneficiaries
count
            return render_template('output.html',
                                   prediction=round(prediction),
                                   graph=graph,
                                   beneficiaries_count=beneficiaries_count)
```

```python
                        return render_template('output.html',
                           prediction=None,
                           graph=None,
                           beneficiaries_count=None)
    return render_template('index.html')

if __name__ == '__main__':
   app.run(debug=True)
```

INDEX.HTML:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home - Unemployed Insurance Beneficiary Forecast</title>
  <style>
    /* General Reset */
    body {
      margin: 0;
      font-family: Arial, sans-serif;
      background-color: #f4f4f9;
      color: #333;
      line-height: 1.6;
    }
    /* Navigation Bar */
    .navbar {
      display: flex;
      justify-content: space-between;
      align-items: center;
      background-color: #007bff;
      padding: 10px 20px;
      color: white;
      position: fixed;
      top: 0;
      width: 100%;
      z-index: 1000;
      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    }

    .navbar a {
      text-decoration: none;
```

```
                    color: white;  margin: 0 15px;  font-size: 18px;
}

.navbar a:hover {
   text-decoration: underline;
}

/* Header Section */
.header {
   text-align: center;
   padding: 100px 20px;
   background: linear-gradient(135deg, #007bff, #6c63ff);
   color: white;
}

.header h1 {
   font-size: 2.5rem;
   margin-bottom: 20px;
}

.header a {
   display: inline-block;
   margin-top: 20px;
   padding: 10px 20px;
   font-size: 18px;
   color: #007bff;
   background-color: white;
   text-decoration: none;
   border-radius: 5px;
   transition: 0.3s;
}

.header a:hover {
   background-color: #f4f4f9;
}

/* Footer Section */
.footer {
   text-align: center;
   padding: 20px;
   background-color: #333;
   color: white;
   margin-top: 20px;
```

```
                                   }

        /* Scroll-to-Top Button */
        #scrollUpBtn {
            position: fixed;
            bottom: 20px;
            right: 20px;
            background-color: #007bff;
            color: white;
            border: none;
            padding: 10px 15px;
            border-radius: 50%;
            font-size: 18px;
            cursor: pointer;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
            display: none;
            transition: 0.3s;
        }
        #scrollUpBtn:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <!-- Navigation Bar -->
    <div class="navbar">
        <div class="nav-left">
            <a href="#">Home</a>
            <a href="#services">Services</a>
            <a href="#contact">Contact</a>
        </div>
    </div>
    <!-- Header Section -->
    <div class="header">
        <h1>Welcome to the Unemployed Insurance Beneficiary Forecast App</h1>
        <p>Plan and predict better with our intelligent forecasting tool.</p>
        <a href="{{ url_for('inspect') }}">Get Started</a>
    </div>

    <!-- Services Section -->
    <section id="services" style="padding: 50px 20px;">
        <h2 style="text-align: center; margin-bottom: 20px;">Our Services</h2>
        <p style="text-align: center; max-width: 600px; margin: 0 auto;">
```

```
            We offer precise and reliable forecasts for unemployment
benefits, helping you make better financial decisions. Explore our services and discover how we
can assist you.
        </p>
    </section>
    <!-- Contact Section -->
    <section id="contact" style="padding: 50px 20px; background-color: #f4f4f9;">
        <h2 style="text-align: center; margin-bottom: 20px;">Contact Us</h2>
        <p style="text-align: center; max-width: 600px; margin: 0 auto;">
            Have questions? Feel free to reach out to us at
<strong>support@forecastapp.com</strong>.
        </p>
    </section>
    <!-- Footer Section -->
    <div class="footer">
        <p>&copy; 2024 Unemployed Insurance Forecast App. All rights reserved.</p>
    </div>
    <!-- Scroll-to-Top Button -->
    <button id="scrollUpBtn" onclick="scrollToTop()">↑</button>
    <!-- Scroll-to-Top Script -->
    <script>
        // Scroll-to-Top Button
        const scrollUpBtn = document.getElementById("scrollUpBtn");
        window.onscroll = function () {
            if (document.body.scrollTop > 100 || document.documentElement.scrollTop > 100) {
                scrollUpBtn.style.display = "block";
            } else {
                scrollUpBtn.style.display = "none";
            }
        };
        function scrollToTop() {
            window.scrollTo({ top: 0, behavior: "smooth" });
        }
    </script>
</body>
</html>
```

INSPECT.HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
                <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Beneficiary Forecast Inspection</title>
  <style>
    /* General Reset */
    body {
      margin: 0;
      font-family: Arial, sans-serif;
      background-color: #f4f4f9;
      color: #333;
    }

    /* Navigation Bar */
    .navbar {
      display: flex;
      justify-content: space-between;
      align-items: center;
      background-color: #007bff;
      padding: 10px 20px;
      color: white;
      position: fixed;
      top: 0;
      width: 100%;
      z-index: 1000;
      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    }
    .navbar a {
      text-decoration: none;
      color: white;
      margin: 0 15px;
      font-size: 18px;
    }

    .navbar a:hover {
      text-decoration: underline;
    }

    /* Hero Section */
    .hero {
      background: linear-gradient(135deg, #007bff, #6c63ff);
      color: white;
      text-align: center;
      padding: 100px 20px;
```

```
                    }

.hero h1 {
    font-size: 2.5rem;
    margin-bottom: 20px;
}

.hero p {
    font-size: 1.2rem;
    margin-bottom: 30px;
}

.hero img {
    width: 100%;
    max-width: 500px;
    margin: 20px auto;
    border-radius: 10px;
}

/* Form Section */
.form-section {
    padding: 50px 20px;
    max-width: 800px;
    margin: 0 auto;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

.form-section h2 {
    text-align: center;
    margin-bottom: 20px;
    color: #007bff;
}

.form-section label {
    display: block;
    margin-bottom: 10px;
    font-weight: bold;
}

.form-section input, .form-section button {
    width: 100%;
```

```
                                        padding: 10px;
    margin-bottom: 20px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.form-section button {
    background-color: #007bff;
    color: white;
    border: none;
    cursor: pointer;
    transition: 0.3s;
}

.form-section button:hover {
    background-color: #0056b3;
}

/* Footer Section */
.footer {
    text-align: center;
    padding: 20px;
    background-color: #333;
    color: white;
    margin-top: 20px;
}

/* Images Grid Section */
.images-section {
    padding: 50px 20px;
    background-color: #f9f9f9;
}
.images-section h2 {
    text-align: center;
    margin-bottom: 30px;
    color: #007bff;
}
.images-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
    max-width: 1200px;
```

```
                                          margin: 0 auto;
        }

        .images-grid img {
            width: 100%;
            border-radius: 10px;
            box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>
<body>
    <!--Navigation Bar -->
    <div class="navbar">
        <a href="#">Home</a>
        <a href="#form">Forecast Inspection</a>
        <a href="#images">Project Images</a>
        <a href="#contact">Contact</a>
    </div>
         <!-- Hero Section    -->
       <div class="hero">
        <h1>Beneficiary Forecast Inspection</h1>
        <p>Discover insights and predictions for unemployment insurance beneficiaries.</p>
        <img src="https://media.istockphoto.com/id/506517068/photo/house-car-savings-medical-
and-travel-suitcases-on-of-
coins.jpg?s=1024x1024&w=is&k=20&c=Fqja9zUVKaBzT_ZYTRKGBQSthUnnd1S6FVz3QV
J5bS8=" alt="Forecasting Tool">
    </div>
    <!--Form Section -->
    <section id="form" class="form-section">
        <h2>Get Your Forecast</h2>
        <form method="POST" action="{{ url_for('inspect') }}">
            <label for="input_date">Enter Date:</label>
            <input type="date" id="input_date" name="input_date" required>
            <button type="submit">Get Forecast</button>
        </form>
    </section>
   <!--Images Section -->
    <section id="images" class="images-section">
        <h2>Related Project Images</h2>
        <div class="images-grid">
            <img
src="https://media.istockphoto.com/id/892978584/photo/concept.jpg?s=1024x1024&w=is&k=2
0&c=OxfDfLEtiDCmjn_lLcdeCqtAzT27PSZfghEBWZGGKAw=">
```

```
                                   <img src="https://img.freepik.com/premium-photo/young-man-holding-phone-with-insurance-icon_218381-5216.jpg?w=826" alt="Project Image 2">
        <img src="https://img.freepik.com/free-vector/flat-colorful-insurance-infographic-with-manager-diagrams-graphs-health-retirement-life-property-assurance-illustration_1284-51064.jpg?t=st=1732636809~exp=1732640409~hmac=b6a337d5135fdf3f8820897fe3371914963c9c6bab70b61e265f844cb87c5c9c&w=740">
        <img src="https://miro.medium.com/v2/resize:fit:1200/1*eZW5iyCIPj0U9HHUELU79w.png">
      </div>
    </section>
   <!--Footer Section-->
    <div id="contact" class="footer">
      <p>&copy; 2024 Beneficiary Forecast App. All rights reserved.</p>
      <p>Contact us: <a href="mailto:support@forecastapp.com" style="color: #4dafff;">support@forecastapp.com</a></p>
    </div>
</body>
</html>
```

OUTPUT.HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Prediction Output</title>
   <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
   <style>
      /* General Reset */
      body {
         margin: 0;
         font-family: 'Arial', sans-serif;
         background: linear-gradient(135deg, #1e3c72, #2a5298);
         color: white;
      }
      /* Page Header */
      header {
         text-align: center;
         padding: 20px;
         background-color: rgba(0, 0, 0, 0.7);
         box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2);
         position: sticky;
         top: 0;
```

```
            z-index: 1000;
}

header h1 {
    margin: 0;
    font-size: 2.5rem;
    color: #00ffcc;
    text-shadow: 0 0 20px rgba(0, 255, 204, 0.8);
}
/* Content Section */
.content {
    max-width: 900px;
    margin: 50px auto;
    padding: 20px;
    background: rgba(255, 255, 255, 0.1);
    border-radius: 10px;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.3);
}
.content h2 {
    font-size: 1.8rem;
    color: #ffe259;
    text-shadow: 0 0 10px rgba(255, 226, 89, 0.8);
}
.content p {
    font-size: 1.5rem;
    margin: 10px 0;
    padding: 10px;
    color: #ffffff;
    background: rgba(0, 0, 0, 0.4);
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
}
p.prediction-value {
    color: #1e90ff;
    font-weight: bold;
    font-size: 1.8rem;
    text-shadow: 0 0 10px rgba(30, 144, 255, 0.8);
}
p.beneficiaries-value {
    color: #32cd32;
    font-weight: bold;
    font-size: 1.8rem;
    text-shadow: 0 0 10px rgba(50, 205, 50, 0.8);
```

```
                    }

/* Graph Container */
.graph-container {
    margin-top: 30px;
    text-align: center;
}

.graph-container h2 {
    font-size: 1.8rem;
    color: #ff6f61;
    text-shadow: 0 0 10px rgba(255, 111, 97, 0.8);
}
.graph-container div {
    margin: 0 auto;
    max-width: 800px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.4);
    border-radius: 10px;
    overflow: hidden;
}
/* Error Message */
.error-message {
    color: #ff0000;
    font-size: 1.5rem;
    text-align: center;
    background: rgba(255, 0, 0, 0.1);
    padding: 10px;
    border-radius: 5px;
    margin-bottom: 20px;
    box-shadow: 0 2px 10px rgba(255, 0, 0, 0.3);
}

/* Footer Section */
footer {
    text-align: center;
    padding: 20px;
    background: rgba(0, 0, 0, 0.8);
    color: white;
    margin-top: 40px;
    font-size: 0.9rem;
    border-top: 1px solid rgba(255, 255, 255, 0.2);
}
footer a {
```

```
                    color: #00ffcc;
            text-decoration: none;
        }

        footer a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <!-- Page Header -->
    <header>
        <h1>Unemployed Insurance Beneficiary Forecast Results</h1>
    </header>
    <!-- Content Section -->
    <div class="content">
        <!-- Error Message -->
        {% if error %}
            <div class="error-message">
                {{ error }}
            </div>
        {% endif %}
        <!-- Display Prediction -->
        {% if prediction %}
            <h2>Predicted Value:</h2>
            <p class="prediction-value">{{ prediction }}</p>
        {% endif %}
        <!-- Display Beneficiaries Count -->
        {% if beneficiaries_count is not none %}
            <h2>Total Beneficiaries Count:</h2>
            <p class="beneficiaries-value">{{ beneficiaries_count }}</p>
        {% endif %}
        <!-- Display Graph -->
        {% if graph %}
            <div class="graph-container">
                <h2>Forecast Graph:</h2>
                <div>{{ graph | safe }}</div>
            </div>
        {% endif %}
    </div>
    <!-- Footer Section -->
    <footer>
        <p>&copy; 2024 Beneficiary Forecast App. All rights reserved.</p>
```

```
                                        <p>Contact us: <a
<href="mailto:support@forecastapp.com">support@forecastapp.com</a></p>
    </footer>
</body>
</html>
```