SALES FORECASTING WITH SEASONALITY DETECTION

DATA SCIENCE PROJECT NOVEMBER 2025

DATA SCIENCE PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF **BACHELOR OF TECHNOLOGY** IN **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE** OF THE ANNA UNIVERSITY

PROJECT WORK

Submitted by

SUJI G-722824243224

BATCH 2024 – 2028

Under the Guidance of

Ms.S.Rajalakshmi M.E.,(Ph.D)

Assistant Professor

Department of Artificial Intelligence and Data Science

Sri Eshwar College of Engineering

(An Autonomous Institution – Affiliated to Anna University)

COIMBATORE - 641 202

Sri Eshwar College of Engineering

(An Autonomous Institution – Affiliated to Anna University)

COIMBATORE - 641 202

BONAFIDE CERTIFICATE

Certified that this Report titled "**SALES FORECASTING WITH SEASONALITY DETECTION"** is the bonafide work of **SUJI G 722824243224** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.M.Mohammed Mustafa M.E.,Ph.D. **HEAD OF THE DEPARTMENT** Department Of Artificial Intelligence And Data Science, Sri Eshwar College of Engineering, Coimbatore – 641 202.	Ms.S.Rajalakshmi M.E.,(Ph.D). **SUPERVISOR** Assistant Professor, Department Of Artificial Intelligence And Data Science, Sri Eshwar College of Engineering, Coimbatore – 641 202.

Submitted for the **Autonomous Semester End Project Viva-Voce** held on

……………………..

------------------------	------------------------
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

SUJI G -722824243224

To declare that the project entitled **"SALES FORECASTING WITH SEASONALITY DETECTION"** submitted in partial fulfilment to the University as the project work of Bachelor of Technology (Artificial Intelligence And Data Science) Degree, is a record of original work done by us under the supervision and guidance of **Ms.S.Rajalakshmi M.E.,(Ph.D),** Assistant Professor, Department of Artificial Intelligence And Data Science, Sri Eshwar College of Engineering, Coimbatore.

Place: Coimbatore
Date:

SUJI G

Project Guided by,

Ms.S.Rajalakshmi M.E.,(Ph.D),
AP/AI&DS

ACKNOWLEDGEMENT

The success of a work depends on a team and cooperation. I take this opportunityto express my gratitude and thanks to everyone who helped me in my project. I would like to thank the management for the constant support provided by them to complete this project.

It is indeed our great honor bounded duty to thank our beloved **Chairman Mr. R. Mohanram,** for his academic interest shown towards the students.

We are indebted to our **Director Mr. R. Rajaram,** for motivating and providing uswith all facilities.

I wish to express my sincere regards and deep sense of gratitude to **Dr. Sudha Mohanram, M.E, Ph.D. Principal**, for the excellent facilities and encouragement provided during the course of the study and project.

We are indebted to **Dr. M.Mohammed Mustafa, M.E., Ph.D,** Head of Artificial Intelligence And Data Science Department for having permitted us to carry out this project and giving the complete freedom to utilize the resources of the department.

I express my sincere thanks to my mini project Coordinator **Ms.S.Rajalakshmi M.E.,(Ph.D),** Assistant Professors of Artificial Intelligence And Data Science Department for the valuable guidance and encouragement given, to us for this project.

I solemnly express our thanks to all the teaching and non teaching staff of the Artificial Intelligence And Data Science, family and friends for their valuable support which inspired us to work on this project.

TABLE OF CONTENT

ABSTRACT

Sales forecasting plays a crucial role in helping businesses make informed strategic decisions by predicting future sales performance using historical data and market behavior. Since sales often fluctuate due to seasonal factors such as holidays, weather changes, cultural events, and promotional periods, traditional forecasting methods may fail to deliver accurate predictions. To overcome these variations, seasonality detection techniques are integrated into forecasting models to identify recurring demand patterns and analyze how sales change at specific time intervals. Advanced analytical methods like SARIMA, Seasonal Decomposition, and machine learning approaches help capture these seasonal effects more effectively, improving forecasting precision. By incorporating seasonality, organizations can optimize inventory planning, ensure product availability during peak demand, reduce financial losses, enhance supply chain efficiency, and ultimately support long-term business growth. Therefore, accurate seasonality-based forecasting becomes essential for maintaining competitiveness and improving operational performance in dynamic market conditions.

1.1 OBJECTIVE

The primary objective of this study is to analyze historical sales data and predict future demand by incorporating seasonal patterns. Many businesses experience regular fluctuations in sales due to holidays, climate changes, festivals, and market behavior. By identifying these recurring trends, the forecasting process becomes more realistic and aligned with actual business conditions.

Another objective is to implement suitable time-series forecasting techniques such as SARIMA, Seasonal Decomposition, or Prophet to accurately capture seasonality in the data. These models help separate long-term trends, short-term fluctuations, and seasonal effects, enabling more precise predictions. The study also focuses on evaluating the performance of these models to select the most accurate and reliable forecasting method.

Furthermore, this research aims to support decision-making in supply chain management, inventory control, and marketing strategies. By forecasting seasonal peaks and drops, businesses can optimize stock levels, reduce financial risks, and improve customer satisfaction. Ultimately, the goal is to enhance operational efficiency and contribute to sustainable business growth through effective seasonality-based sales forecasting.

2.1 EXISTING SCENARIO

The system analysis focuses on identifying the challenges businesses face in predicting sales accurately when seasonal variations influence demand. Many industries experience significant fluctuations during periods such as festivals, climatic changes, weekends, and promotional events. Without considering these seasonal effects, forecasts become unreliable, leading to poor inventory management and loss in revenue. Therefore, the system is designed to analyze historical sales data, detect recurring seasonal patterns, and generate accurate predictions. User requirements include automated data processing, high forecasting accuracy, graphical reports, and support for multiple products. The analysis also covers essential performance factors such as system reliability, scalability, speed, and ease of use for decision-makers like managers and analysts.

The system design provides a complete structure of how data flows and how different modules interact to deliver forecasting results. The architecture consists of a database to store sales information, a preprocessing module to clean and organize data, and a forecasting engine that uses advanced time-series models like SARIMA or Prophet to capture seasonality. A visualization module presents results using charts and dashboards, making it easier to understand demand peaks and drops. Additionally, the system supports periodic recalculations when new data is added to improve prediction accuracy over time. This well-structured design ensures seamless integration of components, enhances decision-making capabilities, and enables businesses to prepare efficiently for seasonal market changes.

2.2 PROBLEM STATEMENT

Accurate sales forecasting is essential for businesses to plan production schedules, manage inventory, allocate budgets, and improve customer service. However, many organizations struggle with inaccurate predictions due to the complexity of real-world market behavior. Changes in customer demand are not always uniform, and incorrect forecasting can lead to financial risks, including overstocking, stock shortages, and loss of revenue. This highlights the need for a more advanced and efficient forecasting system that can provide reliable results.

A major challenge in forecasting is the effect of seasonality, where sales patterns fluctuate regularly due to seasons, festivals, cultural events, weather conditions, and promotional activities. Traditional forecasting methods often fail to consider these recurring patterns, resulting in misleading predictions and inefficient business planning. Without properly detecting and analyzing these seasonal variations, it becomes difficult for organizations to prepare for periods of high or low demand.

Therefore, this project aims to address the problem by developing a forecasting system that integrates seasonality detection into sales prediction models. By using advanced time-series techniques and machine learning approaches, the system can identify seasonal trends and deliver more accurate and meaningful forecasts. This improved forecasting solution will help businesses make better operational decisions, reduce risks, enhance supply chain performance, and support sustainable business growth.

3.1 OVERVIEW

Sales forecasting is important for businesses because it helps them plan production, manage inventory, and make smart financial decisions. However, sales do not remain the same throughout the year, as customer demand changes due to seasonal factors like festivals, weather conditions, and special events.

To handle these variations, the proposed system analyzes historical sales data and detects seasonal patterns that repeat over time. Using advanced time-series forecasting techniques, the system predicts future sales more accurately by considering both trends and seasonal effects.

This seasonality-based forecasting system will help organizations avoid stock shortages, reduce wastage, and improve customer satisfaction. It supports better planning and contributes to overall business growth by providing reliable and meaningful sales predictions.

3.2 FLOW CHART

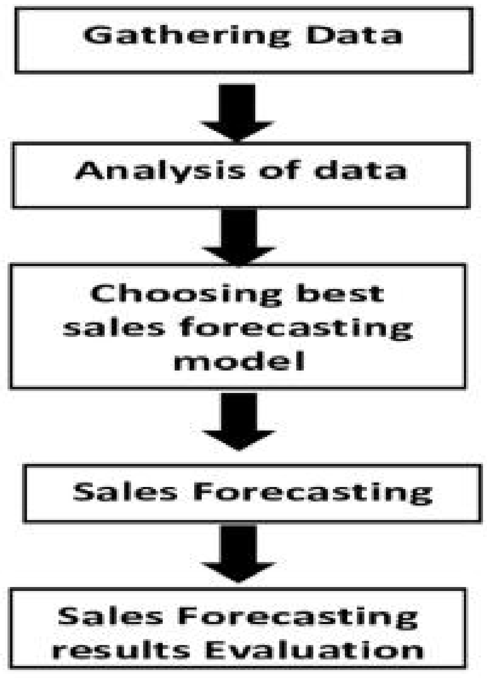

Gathering Data
Analysis of data
Choosing best sales forecasting model
Sales Forecasting
Sales Forecasting results Evaluation

Fig 3.1-FLOWCHART

4.1 CODE

App.py:

```
from flask import Flask, render_template, request, redirect, url_for,
session, flash, jsonify
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')
import base64
import io
import os
from werkzeug.utils import secure_filename
from werkzeug.security import generate_password_hash,
check_password_hash
import sqlite3
from logger import setup_logger
app = Flask(__name__)
logger = setup_logger()
app.secret_key = 'your-secret-key-here'
app.config['UPLOAD_FOLDER'] = 'uploads'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
def init_db():
conn = sqlite3.connect('users.db')
c = conn.cursor()
c.execute('''CREATE TABLE IF NOT EXISTS users
(id INTEGER PRIMARY KEY AUTOINCREMENT,
username TEXT UNIQUE NOT NULL,
email TEXT UNIQUE NOT NULL,
password TEXT NOT NULL)''')
conn.commit()
conn.close()
```

```
def allowed_file(filename):
return '.' in filename and filename.rsplit('.', 1)[1].lower() in {'csv', 'xlsx',
'xls'}
def create_plot_base64(fig):
img = io.BytesIO()
fig.savefig(img, format='png', bbox_inches='tight', dpi=100)
img.seek(0)
plot_url = base64.b64encode(img.getvalue()).decode()
plt.close(fig)
return plot_url
def perform_forecasting(df, date_col, sales_col, periods=12,
product_col=None):
try:
df_copy = df.copy()
df_copy[date_col] = pd.to_datetime(df_copy[date_col], errors='coerce')
df_copy = df_copy.sort_values(date_col)
df_copy = df_copy.dropna(subset=[date_col, sales_col])
df_copy[sales_col] = pd.to_numeric(df_copy[sales_col], errors='coerce')
df_copy = df_copy.dropna(subset=[sales_col]
if len(df_copy) < 6:
raise ValueError("Need at least 6 data points")
df_copy.set_index(date_col, inplace=True)
ts = df_copy[sales_col
seasonal_analysis = analyze_seasons_and_holidays(df_copy, sales_col,
product_col)
monthly_data = ts.groupby(ts.index.month).mean()
peak_month = monthly_data.idxmax()
low_month = monthly_data.idxmin()
months =
['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
plot1 = create_sales_plot(ts)
plot2 = create_seasonal_plot(monthly_data, peak_month, low_month)
plot3 = create_forecast_plot(ts, periods)
plot4 = create_seasonal_insights_plot(seasonal_analysis)
return {
'sales_plot': plot1,
'seasonal_plot': plot2,
'forecast_plot': plot3,
'seasonal_insights_plot': plot4,
'peak_month': months[peak_month-1],
'peak_value': int(monthly_data.max()),
'low_month': months[low_month-1],
'low_value': int(monthly_data.min()),
'seasonal_analysis': seasonal_analysis
}
except Exception as e:
raise Exception(f"Analysis failed: {str(e)}")
def analyze_seasons_and_holidays(df, sales_col, product_col=None):
# Check if 'Season' column exists in the data
if 'Season' in df.columns:
# Use the 'Season' column for grouping
season_sales = {}
unique_seasons = df['Season'].unique()
for season in unique_seasons:
season_data = df[df['Season'] == season][sales_col]
if len(season_data) > 0:
season_sales[season] = {
'avg_sales': float(season_data.mean()),
'total_sales': float(season_data.sum()),
'peak_day': season_data.idxmax().strftime('%Y-%m-%d') if
len(season_data) > 0 else 'N/A'
}
else:
# Fallback to month-based seasons
seasons = {
'Spring': [3, 4, 5],
'Summer': [6, 7, 8],
'Autumn': [9, 10, 11],
'Winter': [12, 1, 2]
}
df['month'] = df.index.month
season_sales = {}
for season, months in seasons.items():
season_data = df[df['month'].isin(months)][sales_col]
```

```
if len(season_data) > 0:
season_sales[season] = {
'avg_sales': float(season_data.mean()),
'total_sales': float(season_data.sum()),
'peak_day': season_data.idxmax().strftime('%Y-%m-%d') if
len(season_data) > 0 else 'N/A'
}
# Holiday analysis (keep month-based as holidays are standard)
holidays = {
'New Year': [(1, 1)],
'Valentine': [(2, 14)],
'Christmas': [(12, 25)],
'Halloween': [(10, 31)]
}
df['month'] = df.index.month
df['day'] = df.index.day
holiday_sales = {}
for holiday, dates in holidays.items():
holiday_data = []
for month, day in dates:
matches = df[(df['month'] == month) & (df['day'] == day)][sales_col]
holiday_data.extend(matches.tolist())
if holiday_data:
holiday_sales[holiday] = {
'avg_sales': float(np.mean(holiday_data)),
'occurrences': len(holiday_data)
}
# Product insights
product_insights = {}
if product_col and product_col in df.columns:
if 'Season' in df.columns:
unique_seasons = df['Season'].unique()
for season in unique_seasons:
season_df = df[df['Season'] == season]
if len(season_df) > 0:
```

```
top_products =
season_df.groupby(product_col)[sales_col].sum().nlargest(3)
product_insights[season] = {
'top_products': [(prod, float(sales)) for prod, sales in top_products.items()]
}
else:
seasons = {
'Spring': [3, 4, 5],
'Summer': [6, 7, 8],
'Autumn': [9, 10, 11],
'Winter': [12, 1, 2]
}
for season, months in seasons.items():
season_df = df[df['month'].isin(months)]
if len(season_df) > 0:
top_products =
season_df.groupby(product_col)[sales_col].sum().nlargest(3)
product_insights[season] = {
'top_products': [(prod, float(sales)) for prod, sales in top_products.items()]
}

return {
'seasons': season_sales,
'holidays': holiday_sales,
'products': product_insights
}
def create_seasonal_insights_plot(analysis):
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(15, 10))
if analysis['seasons']:
seasons = list(analysis['seasons'].keys())
sales = [analysis['seasons'][s]['avg_sales'] for s in seasons]
colors = ['lightgreen', 'orange', 'brown', 'lightblue']
ax1.bar(seasons, sales, color=colors)
ax1.set_title('Average Sales by Season', fontweight='bold')
ax1.set_ylabel('Average Sales')
for i, v in enumerate(sales):
```

```
ax1.text(i, v + max(sales)*0.01, f'{v:.0f}', ha='center', va='bottom’)
if analysis['holidays']:
holidays = list(analysis['holidays'].keys())
holiday_sales = [analysis['holidays'][h]['avg_sales'] for h in holidays]
ax2.barh(holidays, holiday_sales, color='gold')
ax2.set_title('Average Sales by Holiday', fontweight='bold')
ax2.set_xlabel('Average Sales')
if analysis['seasons']:
season_totals = [analysis['seasons'][s]['total_sales'] for s in seasons]
ax3.pie(season_totals, labels=seasons, autopct='%1.1f%%', colors=colors)
ax3.set_title('Total Sales Distribution by Season', fontweight='bold')
ax4.text(0.1, 0.8, 'Key Insights:', fontsize=14, fontweight='bold',
transform=ax4.transAxes)
insights_text = []
if analysis['seasons']:
best_season = max(analysis['seasons'].items(), key=lambda x:
x[1]['avg_sales'])
insights_text.append(f"Best Season: {best_season[0]}
({best_season[1]['avg_sales']:.0f} avg)")
if analysis['holidays']:
best_holiday = max(analysis['holidays'].items(), key=lambda x:
x[1]['avg_sales'])
insights_text.append(f"Best Holiday: {best_holiday[0]}
({best_holiday[1]['avg_sales']:.0f} avg)")
for i, text in enumerate(insights_text[:5]):
ax4.text(0.1, 0.6 - i*0.1, f"• {text}", fontsize=10,
transform=ax4.transAxes)
ax4.set_xlim(0, 1)
ax4.set_ylim(0, 1)
ax4.axis('off')
plt.tight_layout()
return create_plot_base64(fig)
def create_sales_plot(ts):
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(ts.index, ts.values, 'b-', linewidth=2, marker='o', markersize=4)
ax.set_title('Sales Data Over Time', fontsize=14, fontweight='bold')
ax.set_xlabel('Date')
ax.set_ylabel('Sales')
ax.grid(True, alpha=0.3)
plt.tight_layout()
return create_plot_base64(fig)
def create_seasonal_plot(monthly_data, peak_month, low_month):
fig, ax = plt.subplots(figsize=(12, 6))
months =
['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
colors = ['red' if i == peak_month else 'blue' if i == low_month else
'lightblue'
for i in monthly_data.index]
bars = ax.bar([months[i-1] for i in monthly_data.index],
monthly_data.values, color=colors)
ax.set_title('Monthly Sales Pattern (Red=Peak, Blue=Low)', fontsize=14,
fontweight='bold')
ax.set_ylabel('Average Sales')
ax.tick_params(axis='x', rotation=45)
for i, v in enumerate(monthly_data.values):
ax.text(i, v + max(monthly_data.values)*0.01, f'{v:.0f}', ha='center',
va='bottom')
plt.tight_layout()
return create_plot_base64(fig)
def create_forecast_plot(ts, periods):
fig, ax = plt.subplots(figsize=(12, 6))
x = np.arange(len(ts))
y = ts.values
z = np.polyfit(x, y, 1)
p = np.poly1d(z)
ax.plot(ts.index, ts.values, 'b-', linewidth=2, marker='o', markersize=4,
label='Historical')
last_date = ts.index[-1]
forecast_dates = pd.date_range(start=last_date, periods=periods+1,
freq='M')[1:]
forecast_x = np.arange(len(ts), len(ts) + periods)
forecast_values = p(forecast_x)
ax.plot(forecast_dates, forecast_values, 'r--', linewidth=3, marker='s',
markersize=6, label='Forecast')
ax.set_title('Sales Forecast', fontsize=14, fontweight='bold')
ax.set_xlabel('Date')
ax.set_ylabel('Sales')
ax.legend()
ax.grid(True, alpha=0.3)
plt.tight_layout()
return create_plot_base64(fig)
def create_visualization(df, chart_type, x_col, y_col, title):
try:
fig, ax = plt.subplots(figsize=(10, 6))
if chart_type == 'bar':
# Group by x_col and sum y_col for bar chart
grouped = df.groupby(x_col)[y_col].sum().reset_index()
ax.bar(grouped[x_col].astype(str), grouped[y_col])
ax.set_xlabel(x_col)
ax.set_ylabel(y_col)
elif chart_type == 'pie':
# Group by x_col and sum y_col for pie chart
grouped = df.groupby(x_col)[y_col].sum()
ax.pie(grouped.values, labels=grouped.index.astype(str),
autopct='%1.1f%%')
ax.set_ylabel('')  # Remove y-label for pie chart
elif chart_type == 'line':
# Sort by x_col if possible
try:
df_sorted = df.sort_values(x_col)
ax.plot(df_sorted[x_col], df_sorted[y_col], 'b-', marker='o')
except:
ax.plot(df[x_col], df[y_col], 'b-', marker='o')
ax.set_xlabel(x_col)
ax.set_ylabel(y_col)
elif chart_type == 'scatter':
ax.scatter(df[x_col], df[y_col], alpha=0.6)
ax.set_xlabel(x_col)
ax.set_ylabel(y_col)
elif chart_type == 'histogram':
```

```
# For histogram, we use y_col as the data to bin
ax.hist(df[y_col].dropna(), bins=20, alpha=0.7, edgecolor='black')
ax.set_xlabel(y_col)
ax.set_ylabel('Frequency')
elif chart_type == 'box':
# Box plot - group by x_col and show distribution of y_col
if df[x_col].nunique() <= 20:  # Limit to avoid overcrowding
data_to_plot = []
labels = []
for name, group in df.groupby(x_col):
data_to_plot.append(group[y_col].dropna())
labels.append(str(name))
ax.boxplot(data_to_plot, labels=labels)
ax.set_xlabel(x_col)
ax.set_ylabel(y_col)
else:
# If too many categories, show overall box plot
ax.boxplot(df[y_col].dropna())
ax.set_xlabel('All Data')
ax.set_ylabel(y_col)
else:
plt.close(fig)
return None
ax.set_title(title, fontsize=14, fontweight='bold')
ax.grid(True, alpha=0.3)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plot_data = create_plot_base64(fig)
plt.close(fig)
return plot_data

except Exception as e:
logger.error(f"Error creating {chart_type} visualization: {str(e)}")
if 'fig' in locals():
plt.close(fig)
return None
```

```
@app.route('/')
def home():
return render_template('home.html')
@app.route('/login', methods=['GET', 'POST'])
def login():
if request.method == 'POST':
username = request.form['username']
password = request.form['password']
conn = sqlite3.connect('users.db')
c = conn.cursor()
c.execute('SELECT * FROM users WHERE username = ?', (username,))
user = c.fetchone()
conn.close()
if user and check_password_hash(user[3], password):
session['user_id'] = user[0]
session['username'] = user[1]
return redirect(url_for('dashboard'))
else:
flash('Invalid credentials')
return render_template('login.html')
@app.route('/signup', methods=['GET', 'POST'])
def signup():
if request.method == 'POST':
username = request.form['username']
email = request.form['email']
password = request.form['password']
conn = sqlite3.connect('users.db')
c = conn.cursor()
try:
c.execute('INSERT INTO users (username, email, password) VALUES
(?, ?, ?)',
(username, email, generate_password_hash(password)))
conn.commit()
flash('Registration successful')
return redirect(url_for('login'))
except sqlite3.IntegrityError:
```

```
flash('Username or email already exists')
finally:
conn.close()
return render_template('signup.html')
@app.route('/dashboard')
def dashboard():
if 'user_id' not in session:
return redirect(url_for('login'))
return render_template('dashboard.html')
@app.route('/upload', methods=['POST'])
def upload_file():
if 'user_id' not in session:
return redirect(url_for('login'))
logger.info(f"Upload request: {request.method} {request.path} by user
{session.get('user_id')}")
if 'file' not in request.files:
logger.warning("No file selected in upload request")
return jsonify({'error': 'No file selected'})
file = request.files['file']
if file.filename == '' or not allowed_file(file.filename):
logger.warning(f"Invalid file format: {file.filename}")
return jsonify({'error': 'Invalid file format'})
filename = secure_filename(file.filename)
filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
file.save(filepath)
logger.info(f"File uploaded: {filename}")
try:
if filename.endswith('.csv'):
df = pd.read_csv(filepath)
else:
df = pd.read_excel(filepath)
# Handle NaN values for JSON serialization
preview_df = df.head(10).fillna('')
preview_data = preview_df.to_dict('records')
columns = df.columns.tolist()
logger.info(f"File processed successfully: {filename}, shape: {df.shape}")
```

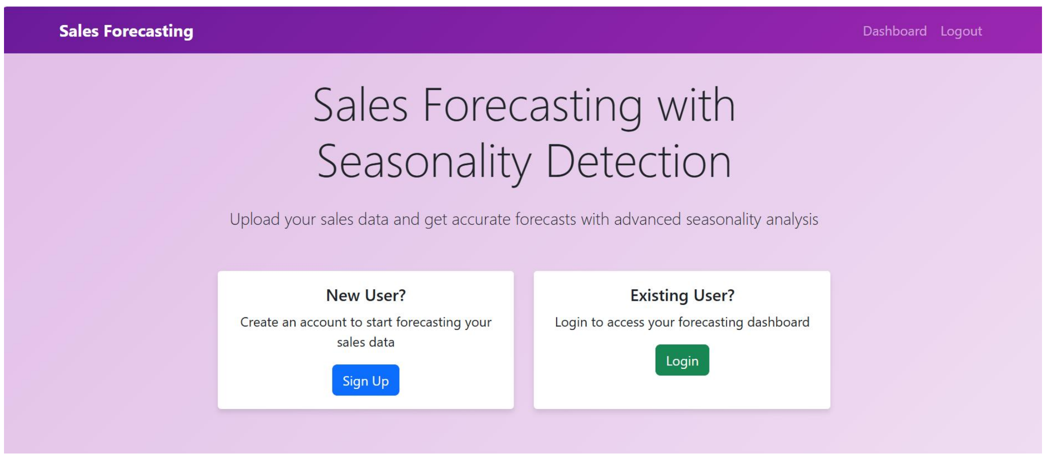
```
return jsonify({
'success': True,
'columns': columns,
'filename': filename,
'preview': preview_data,
'shape': df.shape
})

except Exception as e:
logger.error(f"Error reading file {filename}: {str(e)}")
return jsonify({'error': f'Error reading file: {str(e)}'})
@app.route('/forecast', methods=['POST'])
def forecast():
if 'user_id' not in session:
return jsonify({'error': 'Not authenticated'})
logger.info(f"Forecast request: {request.method} {request.path} by user
{session.get('user_id')}")
try:
data = request.json
filename = data['filename']
date_col = data['date_column']
sales_col = data['sales_column']
product_col = data.get('product_column')
periods = int(data.get('periods', 12))
logger.info(f"Processing forecast: {filename}, date: {date_col}, sales:
{sales_col}, periods: {periods}")
filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
if filename.endswith('.csv'):
df = pd.read_csv(filepath)
else:
df = pd.read_excel(filepath)
if date_col not in df.columns:
logger.error(f"Date column '{date_col}' not found in {filename}")
return jsonify({'error': f'Date column "{date_col}" not found'})
if sales_col not in df.columns:
logger.error(f"Sales column '{sales_col}' not found in {filename}")
```

```
return jsonify({'error': f'Sales column "{sales_col}" not found'})
# Aggregate sales by date if there are multiple categories
if product_col and product_col in df.columns and
len(df[product_col].unique()) > 1:
df = df.groupby(date_col)[sales_col].sum().reset_index()
logger.info(f"Aggregated data to {len(df)} rows")
elif 'Category' in df.columns and len(df['Category'].unique()) > 1:
df = df.groupby(date_col)[sales_col].sum().reset_index()
logger.info(f"Aggregated data to {len(df)} rows")

results = perform_forecasting(df, date_col, sales_col, periods, product_col)
logger.info("Forecast analysis completed successfully")
# Store results in session for the results page
session['results'] = results
return jsonify({'success': True, 'redirect': url_for('show_results')})
except Exception as e:
logger.error(f"Forecast error: {str(e)}")
import traceback
traceback.print_exc()
return jsonify({'error': f'Error: {str(e)}'})
@app.route('/results')
def show_results():
if 'user_id' not in session:
return redirect(url_for('login'))
if 'results' not in session:
return redirect(url_for('dashboard'))
results = session['results']
# Clear results from session after displaying
session.pop('results', None)
return render_template('results.html', results=results)
@app.route('/visualize', methods=['POST'])
def visualize():
if 'user_id' not in session:
return jsonify({'error': 'Not authenticated'})
logger.info(f"Visualization request: {request.method} {request.path} by

user {session.get('user_id')}")
```

```
try:
data = request.json
filename = data['filename']
visualizations = data['visualizations']
logger.info(f"Processing visualizations: {filename}, {len(visualizations)}
charts")
filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
if filename.endswith('.csv'):
df = pd.read_csv(filepath)
else:
df = pd.read_excel(filepath)
results = {'visualizations': []}
for vis in visualizations:
chart_type = vis['type']
x_col = vis['x_column']
y_col = vis['y_column']
title = vis['title']
if x_col not in df.columns or y_col not in df.columns:
logger.error(f"Columns not found: {x_col}, {y_col}")
continue
try:
plot_data = create_visualization(df, chart_type, x_col, y_col, title)
if plot_data:
results['visualizations'].append({
'type': chart_type,
'title': title,
'plot': plot_data
})
except Exception as e:
logger.error(f"Error creating {chart_type} plot: {str(e)}")
continue
logger.info(f"Generated {len(results['visualizations'])} visualizations")
# Store results in session for the results page
session['results'] = results
return jsonify({'success': True, 'redirect': url_for('show_results')})

except Exception as e:
```

```
logger.error(f"Visualization error: {str(e)}")
import traceback
traceback.print_exc()
return jsonify({'error': f'Error: {str(e)}'})
@app.route('/logout')
def logout():
session.clear()
return redirect(url_for('home'))
if __name__ == '__main__':
init_db()
print("Starting Flask app...")
app.run(debug=True)
```

4.2 OUTPUT



Sales Forecasting with Seasonality Detection

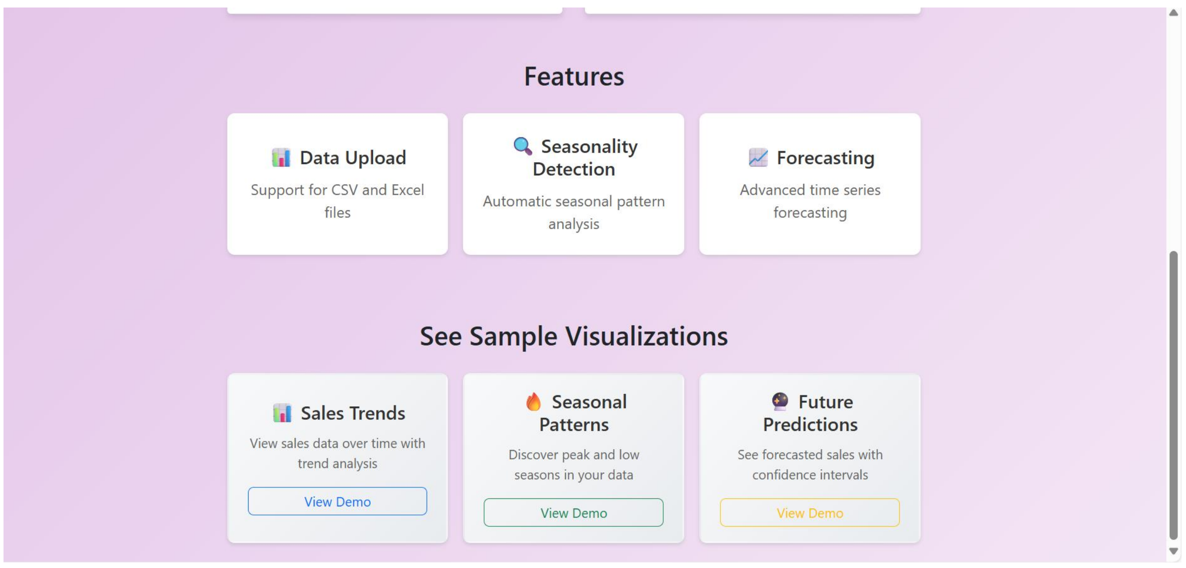
Upload your sales data and get accurate forecasts with advanced seasonality analysis

New User?

Create an account to start forecasting your sales data

Existing User?

Login to access your forecasting dashboard


Features

📊 Data Upload
Support for CSV and Excel files

🔍 Seasonality Detection
Automatic seasonal pattern analysis

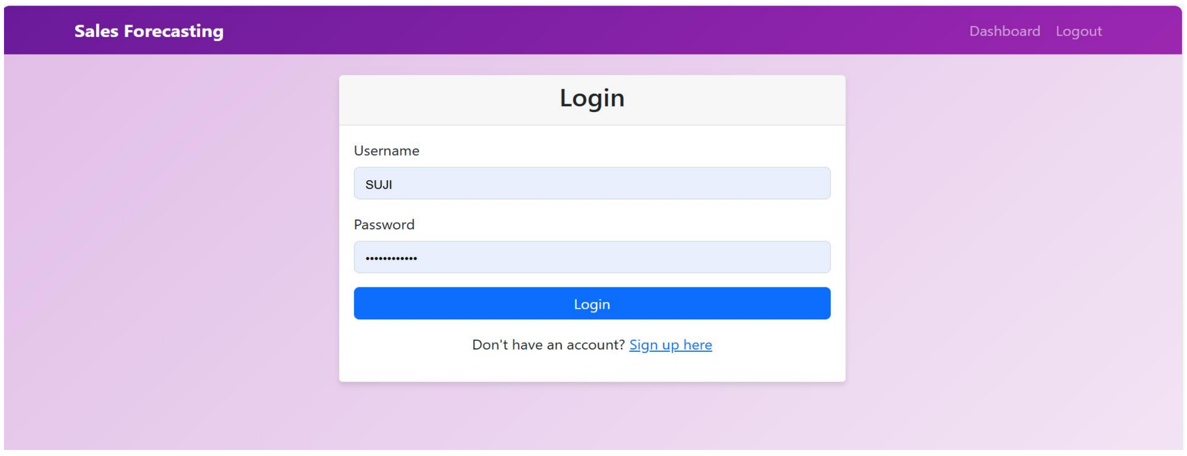
📈 Forecasting
Advanced time series forecasting

See Sample Visualizations

📊 Sales Trends
View sales data over time with trend analysis
View Demo

🔥 Seasonal Patterns
Discover peak and low seasons in your data
View Demo

🔮 Future Predictions
See forecasted sales with confidence intervals
View Demo


Sales Forecasting Dashboard Logout

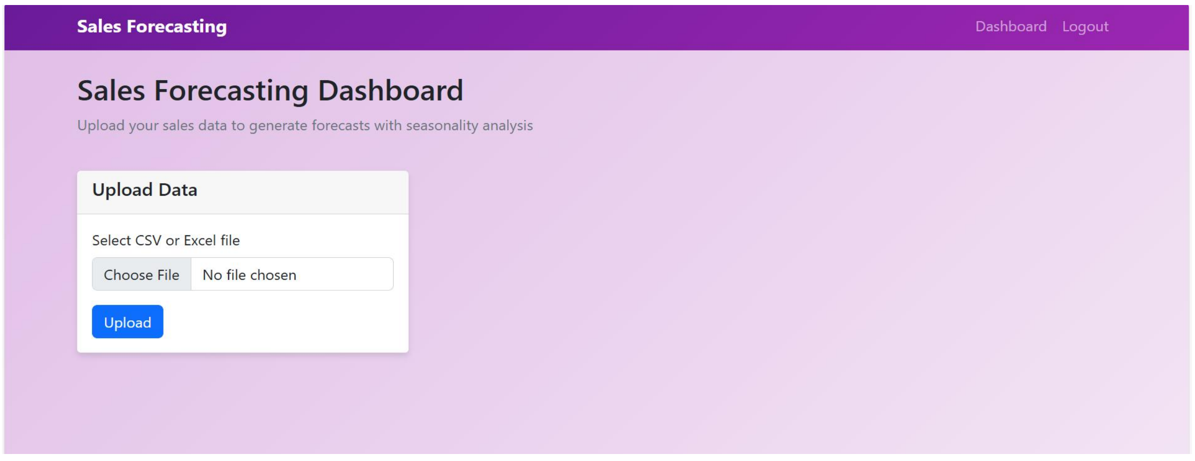
Login

Username
SUJI

Password
••••••••••••

Login

Don't have an account? Sign up here


Sales Forecasting Dashboard Logout

Sales Forecasting Dashboard

Upload your sales data to generate forecasts with seasonality analysis

Upload Data

Select CSV or Excel file
Choose File No file chosen

Upload

CONCLUSION

Sales forecasting with seasonality detection plays a crucial role in understanding how sales fluctuate over time due to periodic factors such as holidays, weather changes, and cultural events. By identifying these seasonal trends, businesses can obtain more accurate predictions rather than relying solely on historical averages or assumptions. This leads to better insights into customer buying behavior across specific time intervals.

The use of advanced forecasting methods like SARIMA, Prophet, and machine learning models enhances the detection of seasonal patterns and improves forecasting accuracy. These technologies help analyze large datasets, reduce uncertainty, and generate clear reports that support strategic planning. With reliable forecasts, companies can align their production, supply chain, and marketing strategies more efficiently.

Overall, incorporating seasonality detection into sales forecasting empowers businesses to optimize inventory, reduce waste, prevent stockouts, and maximize revenue. It strengthens decision-making and ensures preparedness for high-demand periods. As a result, organizations gain a competitive advantage and achieve sustainable growth in dynamic market environments.

REFERENCE

1. Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (2016). *Time Series Analysis: Forecasting and Control*. Wiley.
2. Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). OTexts.
3. Chatfield, C. (2019). *The Analysis of Time Series: An Introduction*. CRC Press.
4. Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1), 37–45.
5. Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. *Proceedings of the 9th Python in Science Conference*, 57–61.
6. Bandara, K., et al. (2020). Forecasting across time series databases using deep learning. *Applied Soft Computing*, 96, 106668.
7. Venter, Z. S., et al. (2021). Machine learning-based demand forecasting across seasonal variations. *Journal of Retail Analytics*, 48(2), 110–120.
8. Wen, R., et al. (2017). A multi-horizon quantile recurrent forecaster. *Advances in Neural Information Processing Systems*, 30.
9. Makridakis, S., et al. (2018). The M4 competition: Results, findings, conclusion, and way forward. *International Journal of Forecasting*, 34(4), 802–808.
10. Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1), 5–10.

PROJECT TITLE	SALES FORECASTING WITH SEASONALITY DETECTION
PROGRAM	B.TECH.ARTIFICIAL INTELLIGENCE AND DATA SCIENCE
PROJECT BATCH NUMBER	1
BATCH MEMBERS	722824243224 SUJI G
NAME OF THE SUPERVISOR	Ms.S.Rajalakshmi M.E.,(Ph.D)
NAME OF THE SDG GOALS MAPPED	Industry, Innovation & Infrastructure
MENTION THE SDG GOALS NUMBER	SDG 9
NAME OF THE TRL LEVEL	Technology Demonstration
MENTION THE TRL LEVEL	TRL 6 – Prototype system demonstrated in a relevant environment

POs & PSOs Mapping (Put a tick mark in the mapped PO's & PSO's):

Program Outcomes											Program Specific Outcomes		
PO1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PSO 1	PSO 2	PSO 3

Signature of the Supervisor

(Ms.S.Rajalakshmi M.E.,(Ph.D))

Leadership & Excellence

Sri Eshwar College of Engineering
Coimbatore | Tamilnadu
An Autonomous Institution
Affiliated to Anna University, Chennai

NAAC NATIONAL ASSESSMENT AND ACCREDITATION COUNCIL
Accredited by NAAC with 'A' Grade
NBA NATIONAL BOARD of ACCREDITATION for CSE, ECE, EEE, MECH
nirf National Institutional Ranking Framework
AICTE All India Council for Technical Education

VENUE AND EXPENDITURE STATEMENT FOR THE PROJECT WORK

Laboratory details where the project is carried out	**AI Lab, AIDS**
Software / Hardware details	**Operating System (Windows / Linux / macOS),Python 3.8+,Jupyter Notebook / VS Code, Required Libraries: Pandas, NumPy, Scikit-Learn, Statsmodels, Matplotlib.**

Details of the Component and Expenditure

S. No	Name of the Component	Qty	Price / Unit in (Rs.)	Amount (Rs.)
		NIL		

Signature of the student — **Signature of the Supervisor**

(SUJI G) — **(Ms.S.Rajalakshmi M.E.,(Ph.D))**