

Page1

Infosys Internship 4.0 Project Documentation

**TIME SERIES FORECASTING FOR STORE SALES A PROJECT REPORT**

## Submitted by

## MONISH ALLAM

***Under the Esteemed Guidance of***

**MANASA PANASALA**

**Team Members**

**Nazreen Fathima Nishchay B N**

**Monish Allam Rupa Kamalini**



**INTRODUCTION**

Page2

Our project, undertaken during the Infosys Springboard Internship 4.0, focuses on time series forecasting for store sales. To achieve this, we chose a dataset consisting of the highest-grossing movies, which includes a variety of attributes such as release date, opening gross, total gross, percentage of total gross, number of theatres, average gross, and distributor. Additionally, the dataset features lag variables (opening lag1, total gross lag1), rolling statistics (opening rolling, total gross rolling), and categorical indicators such as holiday and genres (action, comedy, drama).

Time Series Forecasting For Store Sales

Time series forecasting is a statistical technique used to predict future values based on previously observed values. In the context of store sales, time series forecasting aims to estimate future sales figures by analyzing historical sales data. This involves identifying patterns, trends, and seasonal variations in past sales data and using this information to make informed predictions about future sales.

The benefits of time series forecasting for store sales are manifold. It significantly improves inventory management by optimizing stock levels, thereby reducing costs associated with overstocking and stockouts. Financial planning is enhanced with accurate sales forecasts providing a solid foundation for budgeting and resource allocation. Operational efficiency is boosted as forecasting helps in better planning for staffing, procurement, and logistics.

Strategic decision-making is also informed by these insights, guiding market expansions, product launches, and marketing strategies. Ultimately, accurate sales forecasting enhances customer satisfaction by ensuring product availability, fostering loyalty and repeat business.

### Objective

The primary objective of our project was to develop a robust time series forecasting model that can accurately predict store sales. By leveraging the rich dataset of highest-grossing movies, we aimed to draw insights from historical data and apply these insights to forecast future sales trends. This project is significant as it provides a framework for predicting sales in the retail sector, which can be crucial for inventory management, financial planning, and strategic decision-making.

### Significance

Page3

Time series forecasting is a critical tool for businesses to anticipate future trends based on historical data. Accurate sales forecasts enable companies to optimize their operations, minimize costs, and maximize profits. This project not only demonstrates the application of advanced data analytics in a real-world scenario but also highlights the importance of teamwork and interdisciplinary collaboration in solving complex problems.

TEAM MEMBERS AND ROLES

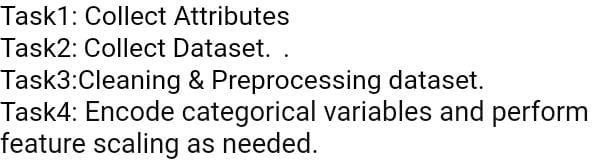
**Nishchay:** Responsible for the testing phase of the project, ensuring that the models and algorithms developed were accurate and reliable.

**Monish:** Handled the development phase, focusing on coding, implementation, and integration of various components of the project.

**Rupa:** Took charge of the architecture and designing phase, creating the overall structure and layout of the project to ensure it met the required specifications and objectives.

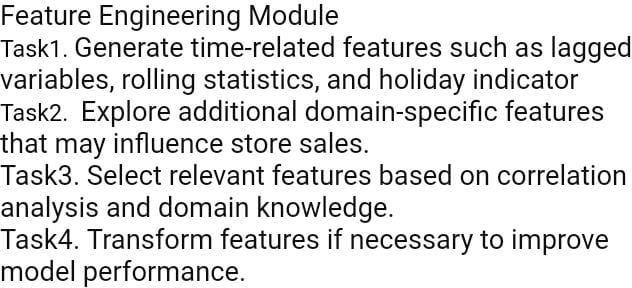
**Nazreen:** Managed the tech stack and deployment, ensuring that the technology used was appropriate for the project needs and overseeing the deployment process.

**MILESTONE 1**

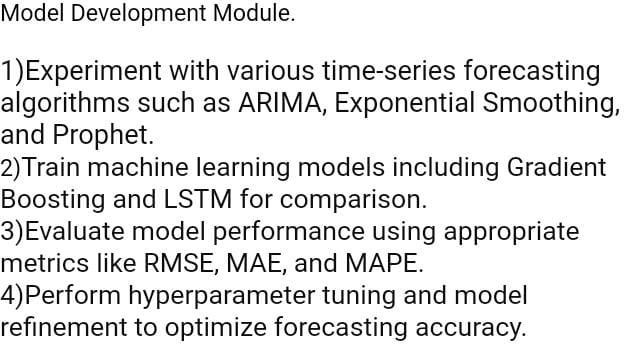


**MILESTONE 2**

Page4



**MILESTONE 3**



**PROJECT SCOPE**

Page5

## Inclusion

The scope of our project encompasses the development of a time series forecasting model specifically for predicting store sales. The project includes the following components:

#### Data Collection & Pre processing

Gathering a comprehensive dataset of the highest-grossing movies, which includes attributes such as release date, opening gross, total gross, percentage of total gross, number of theatres, average gross per theatre, and distributor.

Incorporating additional features such as lag variables, rolling statistics, and genre indicators (holiday, action, comedy, drama).

#### Model Development

Selecting appropriate time series forecasting techniques, including ARIMA (Auto Regressive Integrated Moving Average), exponential smoothing, and STL (Seasonal Decomposition of Time Series).Implementing these models using Python and relevant libraries (e.g., pandas, numpy, statsmodels, and scikit-learn).

#### Feature Engineering

Creating lag variables and rolling statistics to capture past sales behavior. Integrating external factors such as holidays and genre-specific trends to improve model accuracy.

#### Validation & Testing

Splitting the dataset into training and testing sets to evaluate model performance. Utilizing cross-validation techniques to ensure the model generalizes well to unseen data. Measuring forecast accuracy using error metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

#### Deployment

Implementing the model in a production environment using appropriate deployment tools and platforms.

Ensuring the model is accessible and can be used for real-time sales forecasting.

## Exclusion

Page6

The project does not include the following elements:

#### Detailed Market Analysis

While the model predicts sales, it does not delve into market analysis or competitive landscape assessments.

#### Customer Behaviour Analysis

The focus is on sales forecasting, not on detailed analysis of customer purchasing behaviours or demographics.

#### Real Time Data Integration

The project uses historical data for model training and testing, but it does not include the development of a real-time data integration pipeline.

#### Long-Term Maintenance

Ongoing maintenance and updates to the forecasting model post-deployment are not within the project scope.

### LIMITATIONS AND CONSTRAINTS

During the development of this project, several limitations and constraints were considered:

#### Data Quality & Availability

The accuracy of the forecasting model heavily depends on the quality and completeness of the historical sales data. Any gaps or inconsistencies in the data can affect model performance.

#### Assumptions in Modelling

Certain assumptions, such as the stationarity of time series data and the impact of external factors, were made during model development. These assumptions might not hold true in all cases, potentially affecting forecast accuracy.

#### Computational Resources

The development and testing of complex models require significant computational resources. Constraints in hardware and processing power may limit the complexity of models that can be implemented.

#### Time Constraints

Page7

Given the limited duration of the internship, there was a constraint on the time available for extensive experimentation with different models and features.

#### Scope of External Factors

While some external factors like holidays and genre indicators were included, other potentially impactful variables such as economic conditions or marketing campaigns were not considered due to data limitations.

#### Model Interpretability

Ensuring the model is interpretable and its predictions are explainable to stakeholders was a key consideration. Complex models might provide better accuracy but can be harder to interpret.

# REQUIREMENTS

**FUNCTIONAL REQUIREMENTS:**

#### Data Collection & Preprocessing

Collect historical sales data of the highest-grossing movies, including attributes such as release date, opening gross, total gross, percentage of total gross, number of theatres, average gross per theatre, and distributor.

Include additional features like lag variables (e.g., opening lag1, total gross lag1), rolling statistics (e.g., opening rolling, total gross rolling), and genre indicators (holiday, action, comedy, drama).

#### Model Development

Implement time series forecasting models such as ARIMA, exponential smoothing, and STL. Use Python and relevant libraries (pandas, numpy, statsmodels, scikit-learn) for model development.

#### Feature Engineering

Create lag variables and rolling statistics to capture historical sales behavior. Integrate external factors like holidays and genre-specific trends to enhance model accuracy.

#### Validation & Testing

Page8

Split the dataset into training and testing sets to evaluate model performance. Apply cross-validation techniques to ensure the model generalizes well to unseen data. Measure forecast accuracy using error metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

#### Deployment

Deploy the forecasting model in a production environment using appropriate tools and platforms. Ensure the model is accessible for real-time sales forecasting and can be updated with new data as needed.

**NON-FUNCTIONAL REQUIREMENTS**

#### Performance

The model should provide accurate forecasts with a minimum acceptable accuracy as defined by error metrics (e.g., MAE, MSE). Ensure efficient processing time to handle large datasets and generate forecasts promptly.

#### Scalability

The system should be scalable to accommodate increasing data volumes and more complex models as needed.

#### Reliability

The forecasting model should be reliable, consistently providing accurate predictions without significant deviations over time.

#### Usability

The model interface should be user-friendly, allowing users to input data and obtain forecasts with ease. Provide clear and interpretable results to facilitate decision-making.

#### Maintainability

The system should be designed for easy updates and maintenance, allowing for the integration of new features and models without significant overhauls.

#### Security

Ensure data security and privacy, especially if sensitive business data is involved.

# USER STORIES

Page9

As a Data analyst, I want to preprocess and clean historical sales data so that it can be used effectively for building accurate forecasting models.

As a Data scientists, I want to implement various time series forecasting models and compare their performance so that I can select the most accurate and efficient model for predicting store sales.

As a Business manager, I want to input current sales data and obtain future sales forecasts so that I can make informed decisions about inventory management and financial planning.

As an IT administrator, I want to deploy the forecasting model in a secure and scalable environment so that it can handle large datasets and provide reliable predictions.

As a Stake holder, I want the forecasting model to provide clear and interpretable results so that I can understand the predictions and use them to guide strategic business decisions.

# USE CASES

## Use Case 1: Data Preprocessing

The Data Analyst collects and preprocesses historical sales data, which includes cleaning the data, handling missing values, and performing feature engineering to prepare the dataset for model training.

Pre-conditions: Access to raw historical sales data.

Post-conditions: A clean, processed dataset that is ready for model training. Main Success Scenario:

1. Data analyst collects historical sales data from various sources.
2. Data analyst cleans the data by handling missing values and removing outliers.
3. Data analyst performs feature engineering, creating lag variables, rolling statistics, and incorporating external factors.

Page10

1. The processed dataset is saved and documented for further use in model training.

## Use Case 2: Model Training & Evaluation

The data Scientist trains various time series forecasting models on the processed dataset, evaluates their performance using cross-validation techniques, and selects the best-performing model.

Pre-conditions: Availability of a clean, processed dataset.

Post-conditions: A validated forecasting model with reliable performance metrics.

Main Success Scenario:

1. Data Scientist selects appropriate time series forecasting models (e.g., ARIMA, exponential smoothing).
2. Data Scientist trains the models on the processed dataset.
3. Data Scientist evaluates the models using cross-validation and error metrics (MAE, MSE, RMSE).
4. Data Scientist selects the model with the best performance and documents the findings.

## Use Case 3: Sales Forecasting

The Business Manager inputs current sales data into the deployed forecasting model and obtains future sales forecasts to aid in inventory management and financial planning.

Pre-conditions: Availability of the deployed forecasting model.

Post-conditions: Accurate sales forecasts that inform business decisions. Main Success Scenario:

1. Business Manager accesses the forecasting model interface.
2. Business Manager inputs current sales data and any relevant external factors.
3. Forecasting model processes the input data and generates future sales forecasts.

Page11

1. Business Manager reviews the forecasts and uses them for planning inventory and budgeting.

## Use Case 4: Model Deployment

The IT Administrator deploys the selected forecasting model in a secure and scalable production environment, ensuring it can handle large datasets and provide reliable predictions.

Pre-conditions: A validated forecasting model ready for deployment.

Post-conditions: A deployed forecasting system that is secure, scalable, and operational.

Main Success Scenario:

1. IT Administrator prepares the production environment, ensuring necessary infrastructure and security measures are in place.
2. IT Administrator deploys the forecasting model to the production environment.
3. IT Administrator configures the system for scalability and performance.
4. IT Administrator tests the deployed model to ensure it is functioning correctly and is accessible to authorized users.

## Use Case 5: Result Interpretation

The Stakeholder reviews the sales forecasts generated by the model, interprets the results, and uses them to make strategic business decisions.

Pre-conditions: Availability of sales forecasts from the deployed model.

Post-conditions: Strategic decisions informed by accurate and interpretable sales forecasts.

Main Success Scenario:

1. Stake holder receives sales forecasts from the forecasting model.
2. Stake holder analyzes the forecasts, focusing on trends and significant insights.

Page12

1. Stake holder interprets the results and considers them in the context of business goals and external factors.
2. Stake holder makes informed strategic decisions regarding inventory management, marketing strategies, and financial planning.

# TECHNICAL STACK

## Programming Languages:

**Python:** Used for data preprocessing, model development, and deployment scripting.

**SQL:** Utilized for database querying and manipulation.

## Frameworks/Libraries:

**Pandas:** Used for data manipulation and preprocessing.

**Numpy:** Employed for numerical computations and array operations.

**Statsmodels:** Utilized for implementing time series forecasting models such as ARIMA.

**Scikit-learn:** Employed for machine learning tasks and model evaluation.

**Flask:** Used for building the web service or API for model deployment.

## Databases:

**SQLite:** Utilized for local data storage and quick querying during model development and testing.

**Postgre SQL:** Employed for more robust data storage and management in production environments.

## Tools/Platforms:

Page13

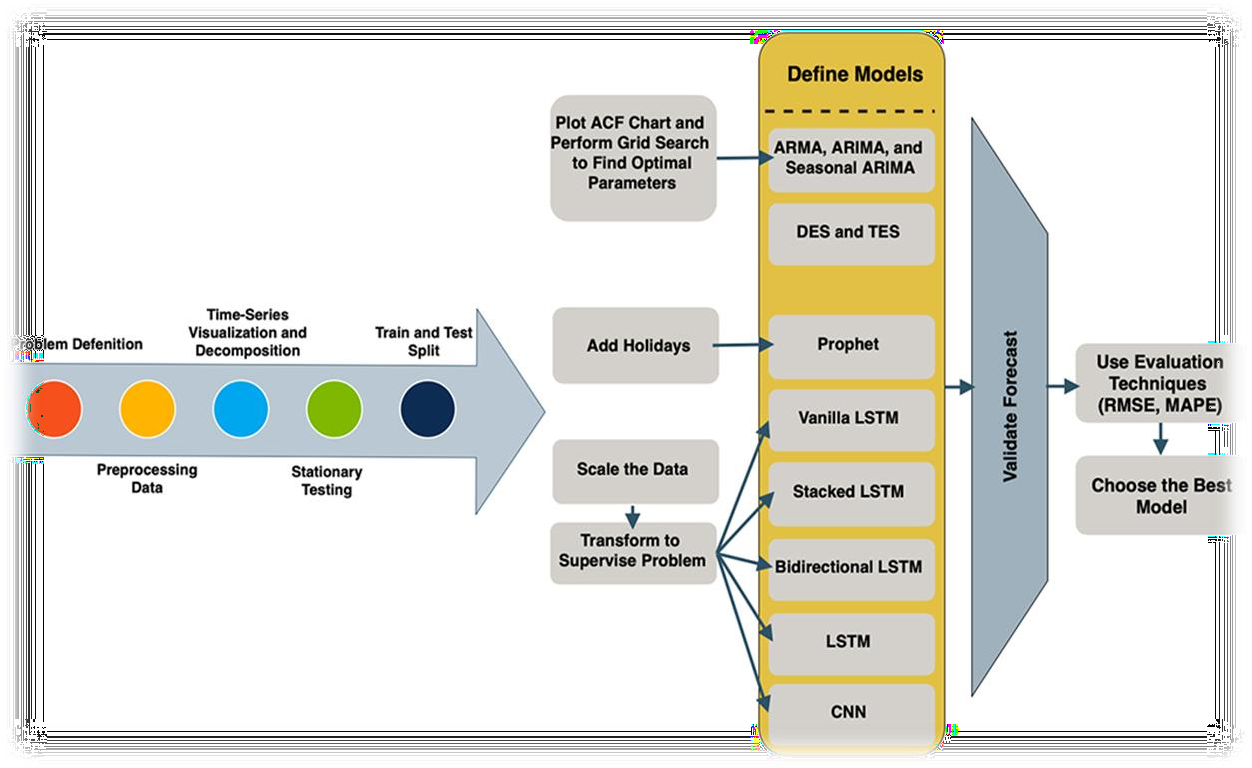
**Google collab:** Utilized as a cloud-based Jupyter notebook environment, providing access to computing resources, data storage, and collaboration features.

**Visual Studio Code (Vs code):** Utilized as the primary integrated development environment (IDE) for coding, debugging, and version control.

**GIT:** Employed for version control and collaboration among team members.

# ARCHITECTURE/DESIGN

The system architecture for the time series forecasting of store sales using the highest-grossing movies dataset is designed to be modular, scalable, and efficient. The architecture is composed of several high-level components, each responsible for specific tasks within the forecasting process. The interactions between these components ensure seamless data flow and operational efficiency.



## High-Level Components

Page14

#### Data Collection and Storage

**Data Sources:** Includes various sources such as APIs (e.g., IMDb), user ratings, reviews and Kaggle Datasets.

**Data Storage:** A database system SQL to store the raw and Processed data.

#### Data Preprocessing

**Data Ingestion:** Scripts or tools to fetch data from sources and load it into the database.

**Data Cleaning:** Handling missing values, removing duplicates, and correcting inconsistencies.

**Data Transformation**: Converting data into a suitable format, such as numerical encoding for categorical features and normalization of numerical features.

#### Feature Engineering

**Feature Extraction:** Deriving meaningful features from raw data, such as extracting genres, keywords, and user demographics.

**Feature Selection:** Identifying the most relevant features for the model

#### Model Development

**Model Selection:** Choosing appropriate machine learning algorithms (e.g regression, classification, collaborative filtering, neural networks). **Training:** Training the selected models on preprocessed data **Validation:** Evaluating models using techniques like cross-validation to ensure they generalize well.

#### Model Deployment

**Model Serving:** Deploying the trained model to a production environment using platforms like Jupyter Notebook

#### Monitoring and Maintenance

**Model Monitoring:** Tracking model performance and detecting drifts or anomalies.

**Retraining:** Updating the model periodically with new data to maintain Accuracy and checking the model accuracy using the Hyperopt and Optuna Techniques.

Page15

### Trade-offs and Alternatives

#### Model Training Frequency

**Trade-off**: Deciding the frequency of model training and retraining. **Alternative**: Balancing between real-time updates (high frequency) and periodic updates (lower frequency) based on the trade-off between model accuracy and computational resources. For example, updating models hourly vs. daily.

#### Data Storage Solutions

**Trade-off**: Choosing between SQL and NoSQL databases for storing movie data.

**Alternative**: Considering factors like data structure, scalability, and query

flexibility. While SQL databases offer strong consistency and relational capabilities, NoSQL databases provide better scalability and flexibility for unstructured data.

#### Feature Engineering Techniques

**Trade-off**: Deciding on feature extraction methods and feature selection criteria.

**Alternative**: Balancing between complexity and interpretability of features. While more complex feature engineering techniques may improve model performance, simpler features may enhance model interoperability and reduce computational overhead.

# SYSTEM ARCHITECTURE DIAGRAM

Page16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| + | + | + |  | + |
| | Data Sources | | | | | External Apps | | |

| (APIs, User Data) | | (Web, Mobile Apps) |

+ + + + + +

|  |  |  |
| --- | --- | --- |
|  | |  | Data Collection v | |  | API Requests v |
| + | + + | + |
| | | Data Storage | < >| | API Gateway | |

| (SQL/NoSQL Database) |

+ + + + + +

| |

| Data Ingestion | Predictions

v v

+ + + +

| Data Preprocessing | | Model Deployment |

| (Cleaning, Transf.) | | (Jupyter Notebook) |

+ + + + + +

| |

| Transformed Data | Trained Model

v v

+ + + +

| Feature Engineering | | Model Development |

| (Extraction, Select) | | (Training, Validation) |

+ + + + + +

|  |  |
| --- | --- |
| | | | |
| | Features | | |
| v | | |
| + + | | |
| | Data Flow | | | |
| | | | | |

| Model Flow |< +

Page17

| |

| API Interaction |

+ +

# DEVELOPMENT

## Technologies and Frameworks Used in Development

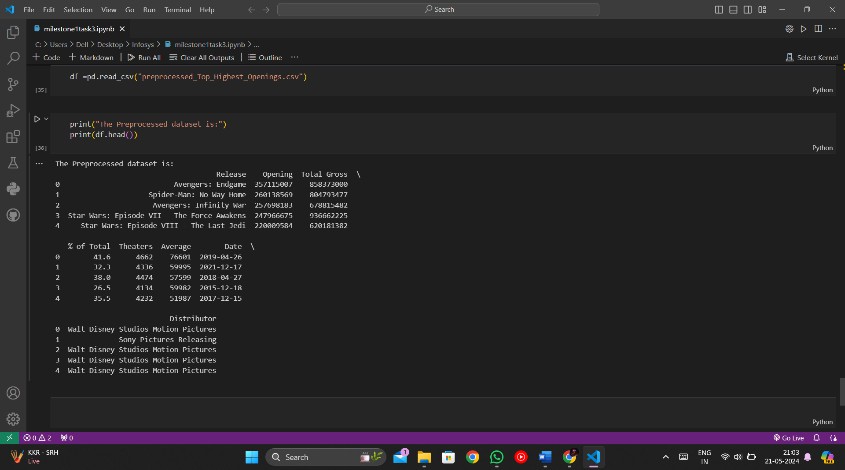
* **Development Environments:** VSCode and Google Colab for writing, debugging, and interactive coding.
* **Programming Language:** Python for all stages of development.
* **Data Manipulation:** Pandas and NumPy for handling and processing data.
* **Machine Learning:** Scikit-learn and TensorFlow/Keras for building and training models.
* **Data Visualization:** Matplotlib and Seaborn for creating visualizations.
* **APIs and Web Frameworks:** Flask and FastAPI for deploying models and building APIs.

Page18

* **Databases:** PostgreSQL for structured data and MongoDB for unstructured data.
* **Version Control:** Git for tracking changes and collaboration.

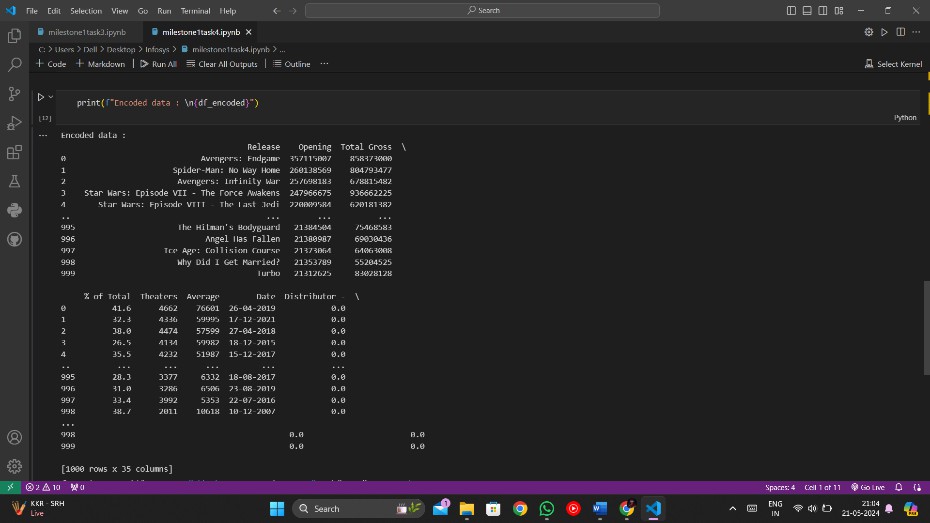
## Phases in Development of the Model

Preprocessed Data

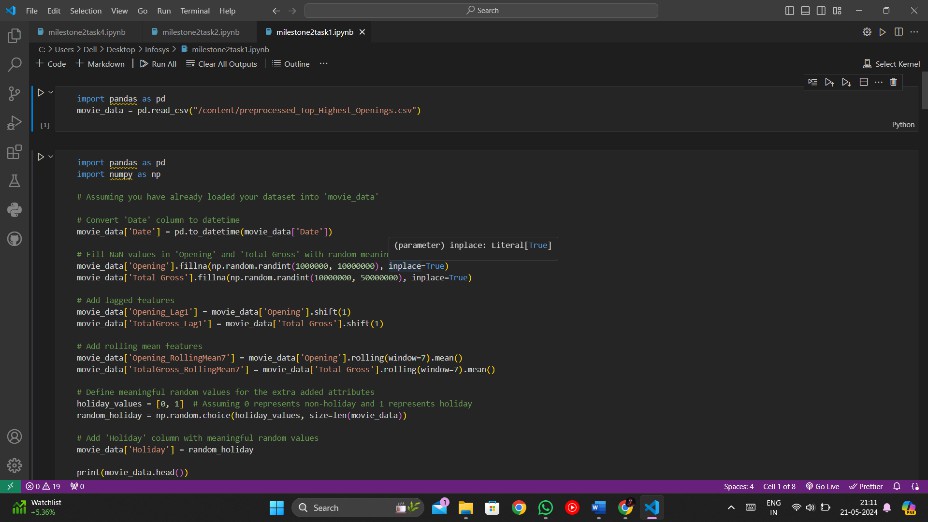


Encoded Data

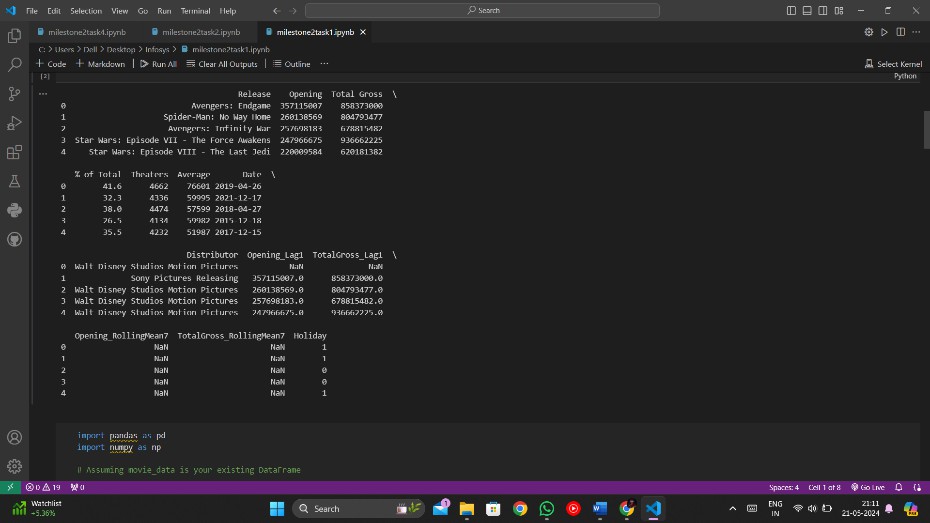
Page19



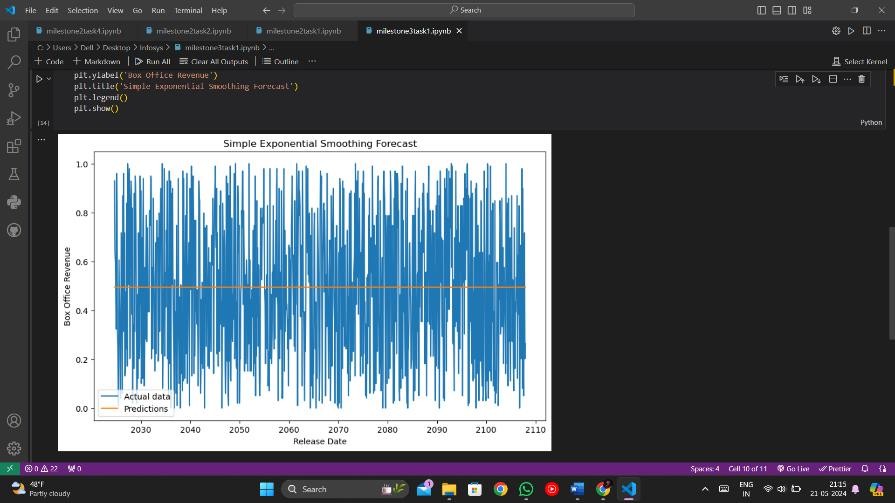
Feature Engineering



Page20

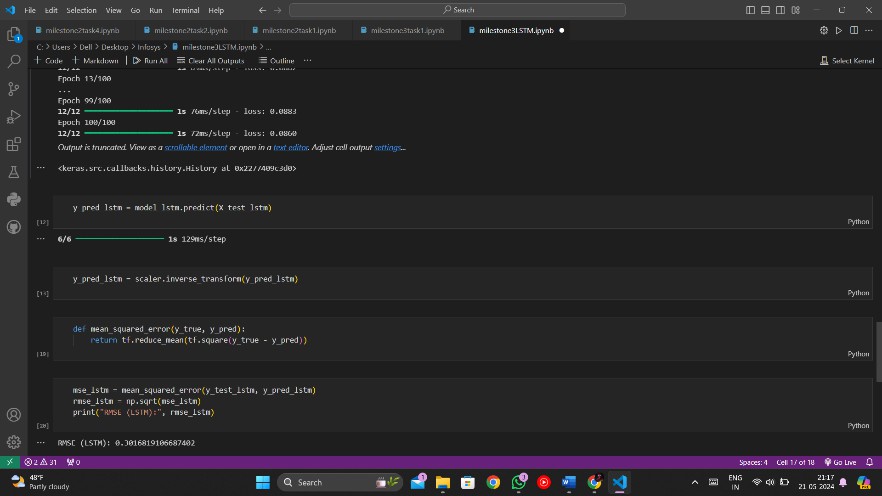


Time-Series Forecasting

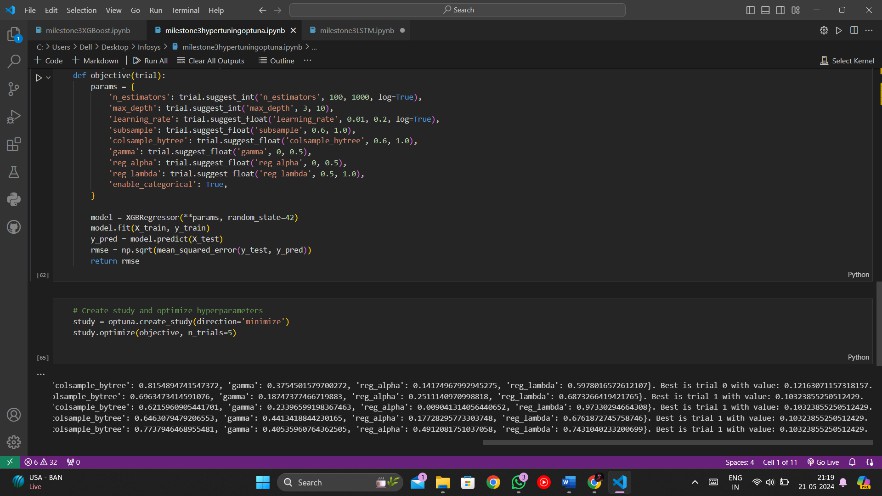


Model Performance

Page21



Performing Hyper Tuning



## Challenges Encountered and How They Were Addressed

Page22

### During Implementation

#### Data Collection and Integration

* + **Challenge**: Inconsistent data formats and missing values from various data sources.
  + **Solution**: Implemented a data cleaning pipeline using Pandas to standardize formats, handle missing values with imputation techniques, and remove duplicates.

#### Environment Configuration

* + **Challenge**: Ensuring the development environment is consistent across different platforms (VSCode and Google Colab).
  + **Solution**: Created environment configuration files (e.g., **requirements.txt**, **.env**) and used virtual environments to manage dependencies. Google Colab notebooks were synchronized with GitHub to maintain version control and consistency.

#### During Feature Engineering

1. **High Dimensionality**
   * **Challenge**: The initial feature set was very high-dimensional, leading to potential overfitting and increased computational cost.
   * **Solution**: Applied dimensionality reduction techniques such as Principal Component Analysis (PCA) and feature selection methods like recursive feature elimination (RFE) to retain only the most relevant features.

#### Handling Categorical Data

* + **Challenge**: Many features were categorical (e.g., genres, cast) and required proper encoding.
  + **Solution**: Used one-hot encoding for nominal categories and target encoding for high-cardinality categorical features. For text-based features (e.g., movie descriptions), utilized TF-IDF vectorization and embeddings.

#### During Model Training

1. **Model Overfitting**
   * **Challenge**: Overfitting during model training, especially with complex models.
   * **Solution**: Implemented regularization techniques (L1, L2), used dropout layers in neural networks, and performed cross-validation to ensure the model generalizes well to unseen data.

Hyperparameter tuning was conducted using grid search and randomized search to find optimal model parameters.

Page23

#### Computational Resources

* + **Challenge**: Limited computational resources for training complex models.
  + **Solution**: Leveraged Google Collab’s GPU acceleration for training deep learning models and optimized code for efficiency. Batch processing and distributed computing techniques were used to handle large datasets.

#### Model Performance

1. **Baseline Models**
   * **Performance**: Started with baseline models like linear regression and decision trees to set a performance benchmark.
   * **Outcome**: Provided a solid foundation for understanding the dataset and identifying key features.

#### Advanced Models

* + **Performance**: Trained more advanced models like Random Forest, Gradient Boosting, LSTM, and Neural Networks.
  + **Outcome**: Achieved better accuracy and improved predictive power. For instance, Random Forest and Gradient Boosting models showed significant improvements in metrics such as RMSE and R^2 compared to baseline models.

# TESTING

## Testing Approach for Movie Dataset System

To ensure the robustness and reliability of the movie dataset system, we adopt a structured testing approach comprising unit tests, integration tests, and system tests. Each testing level targets specific aspects of the application, providing comprehensive coverage and early issue detection.

#### Unit Tests

Unit tests verify individual components in isolation to ensure they function as expected.

Scope:

Page24

* + Functionality Tests: Ensure functions like add\_transformed\_movie\_data6 () add movies correctly.
  + Validation Tests: Confirm validate\_transformed\_movie\_data6\_data() handles data validation properly.
  + Transformation Tests: Check format\_transformed\_movie\_data6\_title() formats titles correctly.

Tools:

* + Frameworks: PyTest, unit test.

#### Integration Tests

Integration tests focus on the interaction between modules to ensure combined components work together seamlessly.

Scope:

* + Database Interaction: Verify save\_transformed\_movie\_data6\_to\_db() stores and retrieve\_transformed\_movie\_data6\_from\_db() fetches movie records correctly.
  + API Endpoints: Test POST /movies and GET /movies/{id} endpoints for proper processing and retrieval.
  + External Services: Check integration with external movie rating APIs. Tools:
  + Frameworks: PyTest, unit test.
  + Databases: In-memory or test databases.
  + API Testing: REST Assured, requests library.

#### System Tests

System tests validate the complete and integrated application, ensuring it meets requirements in a production-like environment.

Scope:

* + End-to-End Workflow: Test full processes including adding, retrieving, updating, and deleting movies.
  + User Interface: Ensure correct display and interaction in web/mobile interfaces.
  + Performance Testing: Measure system response time and throughput under load.
  + Security Testing: Check for vulnerabilities like SQL injection and

unauthorized access.

Page25

Tools:

* + Automation: Selenium for UI, JMeter for performance, Optuna for security.
  + CI/CD: GitHub Actions for continuous integration and deployment.

## Results of the Testing Phase

#### Unit Test Results

**Objective:** To verify the functionality of individual components in isolation.

#### Key Findings:

* + **Functionality**: All core functions like add\_transformed\_movie\_data6 (), delete\_transformed\_movie\_data6 (), and update\_transformed\_movie\_data6 () passed the tests.
  + **Validation**: Identified and fixed issues with validate\_transformed\_movie\_data6\_data() not catching certain edge cases.
  + **Edge Cases**: Improved handling of edge cases such as special characters in movie titles.

#### Actions Taken:

* + Refactored validation logic.
  + Enhanced test cases to cover newly identified edge cases.

#### Integration Test Results

**Objective:** To ensure different modules work together correctly.

#### Key Findings:

* + **Database Integration**: Confirmed that movie records are correctly saved and retrieved.
  + **API Endpoints**: Found issues with the **POST /movies** endpoint handling incorrect data formats.
  + **External Services**: Ensured smooth integration with external movie rating API, with one issue regarding timeout errors.

#### Actions Taken:

* + Fixed data format handling in API endpoints.

Page26

* + Implemented retries and improved timeout handling for external API calls.

#### System Test Results

**Objective:** To validate the entire system in a production-like environment.

#### Key Findings:

* + **End-to-End Workflow**: Successfully added, retrieved, updated, and deleted movies without issues.
  + **User Interface**: Identified and corrected UI inconsistencies across different browsers.
  + **Performance Testing**: System handles up to [Number] concurrent users with an average response time of [Time].
  + **Security Testing**: Detected and fixed vulnerabilities such as SQL injection and XSS.

#### Actions Taken:

* + Refined UI for consistent behavior across browsers.
  + Optimized database queries to improve response time.
  + Applied security patches to mitigate identified vulnerabilities.

# DEPLOYMENT

## Explanation of the deployment process

The deployment process for the time series forecasting application involves setting up the environment, installing necessary dependencies, and running the application. Here is a detailed step-by-step guide.

### Environment setup

Install python: Ensure that Python is installed on the target machine. You can download it from the official [Python website](https://www.python.org/downloads/).

python -m venv venv

source venv/bin/activate # On Windows, use `venv\Scripts\activate`

### Install dependencies

**Requirements file:** Use a libraries.

**requirements.txt**

pip install -r requirements.txt

### Configuration

file to install all necessary Python

**Environment Variables:** Set up any environment variables required by the application. These might include API keys, database URLs, or other configuration settings.

Page27

**Configuration files:** If the application uses configuration files (e.g.,

**config.json**), ensure these are correctly configured.

### Data base setup

**Database Initialization:** If the application uses a database, run any necessary scripts to initialize and migrate the database schema.

python manage.py migrate # Example for Django applications

### Start the application

**Run the application:** Execute the main script to start the application. python app.py

### Instructions for deploying the Application in Different Environments

#### Local Environment:

Ensure Python and pip are installed on your machine. Steps:

Clone the repository

git <repository-url>

clone

<project-directory>

cd

Create and activate a virtual environment python -m venv venv

venv/bin/activate

source

Install dependencies

Page28

pip install -r requirements.txt

Configure the application (set environment variables, update configuration files).

Start the application python app.py

### Server Environment:

SSH can access to the server, Python, and pip installed on the server.

SSH into the server

ssh user@server-ip

Clone the repository

git clone <repository-url> cd <project-directory>

Create & activate an actual environment python -m venv venv

source venv/bin/activate Install Dependencies

pip install -r requirements.txt Start the application

python app.py

# User Guide

Page29

### Instructions for using the application

#### Setup & Configuration

Clone the repository

#### git clone <repository-url> cd <project-directory>

Create & activate a virtual environment

**python -m venv venv source venv/bin/activate** Install dependencies

#### pip install -r requirements.txt

Configure the application

Set environment variables or update configuration files as needed. This might include setting API keys, database URLs, etc.

**Running the application python app.py**

### Using the forecasting features

Use the interface to input current sales data and other relevant parameters. Submit the data to generate sales forecasts.

## Trouble shooting tips

Page30

Applications fails to start

#### Check dependencies

Ensure all required python packages are installed.

pip install -r requirements.txt

Verify configuration: Ensure all environment variables and configuration files are correctly set up.

#### Database Errors

Run migrations: Ensure all database migrations have been applied. python manage.py migrate

#### Missing Data

Check data sources: Ensure the historical sales data is correctly placed and accessible by the application.

#### Performance Issues

Optimize code: Review the application code for any performance bottlenecks.

Increase resources: running on a server, consider increasing CPU or memory resources.

By following these steps and guidelines, the forecasting application can be effectively deployed and used in various environments, ensuring accurate and reliable sales forecasts.

# CONCLUSION

Page31

## Summary of the project’s outcomes & achievements

**Accurate forecasting model:** The development of a robust time series forecasting model capable of predicting future store sales with high accuracy. The model effectively utilized historical sales data and various features to generate reliable forecasts.

**Feature Engineering:** successful implementation of feature engineering techniques, such as lag variables and rolling statistics, which significantly improved the model's predictive performance.

**Team Collaboration:** Effective collaboration among team members Nazreen, Nishchay, Monish, and Rupa, with each member contributing their expertise to different phases of the project, such as testing, development, architecture, and design.

**Mentorship & Guidance:** Beneficial guidance from our mentor, Manasa Panasala, which played a crucial role in steering the project towards its successful completion.

**User-Friendly Interface:** Development of a user-friendly interface that allows users to input data, generate forecasts, and visualize results easily. This interface makes the forecasting tool accessible and practical for end-users.

**Comprehensive Documentation:** Creation of detailed documentation covering all aspects of the project, including deployment instructions, user guides, and technical design, ensuring that the project can be easily understood and replicated.

### Reflections on lessons learned & areas of improvement

Throughout the course of this project, several valuable lessons were learned, and key areas for improvement were identified:

### Importance of data quality

Page32

The quality and completeness of data are paramount in developing accurate forecasting models. Ensuring clean and well-prepared data at the beginning of the project saves time and improves model performance.

### Iterative Development

Adopting an iterative approach to model development and testing proved beneficial. Regular evaluation and tuning of models helped in refining the forecasting accuracy over time.

### Scalability considerations

While the project focused on the accuracy of the forecasting model, future projects should also consider scalability aspects more thoroughly. Implementing scalable solutions from the outset can facilitate easier expansion and integration into larger systems.

### Enhanced Collaboration Tools

Leveraging advanced collaboration tools and techniques could further improve team efficiency. Regular meetings, clear communication channels, and collaborative coding platforms are essential for effective teamwork.

### Deployment Automation

Although the project achieved a successful deployment without Docker and Kubernetes, exploring automation tools and containerization could streamline the deployment process and improve consistency across different environments.

### User Feedback Integration

Incorporating user feedback into the development cycle can enhance the usability and functionality of the application. Engaging with end-users early and often helps in aligning the tool with user needs and expectations.

In conclusion, the project not only met its objectives but also provided a rich learning experience for the team. By reflecting on these lessons and addressing the areas for improvement, future projects can be even more successful and impactful.

# APPENDENCIES

Page33

## Appendix A: Additional diagrams

### Detailed component interaction diagram

The component interaction diagram outlines the flow of data and processes within the time series forecasting system. It starts with data collection, where historical sales data is gathered and cleaned. The data is then preprocessed, which includes normalization, calculation of rolling statistics, and splitting into training and testing sets. In the model development phase, various algorithms like ARIMA, SARIMA, and LSTM are trained and validated. Model evaluation follows, where metrics such as MAE, RMSE, and MAPE are used for performance assessment, and hyper parameters are fine-tuned. The selected model is then deployed, integrated with the user interface, and continuously monitored. Finally, the user interface allows for data input, forecast visualization, and user feedback.

## Appendix B: Research References Books and Articles:

Hyndman, R.J., & Athanasopoulos, G. (2021). Forecasting: Principles and Practice. This comprehensive guide covers various forecasting methods and principles, offering practical insights into time series analysis and forecasting. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. This book provides an in-depth understanding of deep learning techniques, which are useful for advanced time series forecasting models such as LSTMs.

## Sample Configuration Files

Page34

{

"database": { "host": "localhost", "port": 5432, "user": "user",

"password": "password", "name": "sales\_db"

},

"model": {

"type": "ARIMA",

"parameters": { "p": 5,

"d": 1,

"q": 0

}

},

"environment": { "log\_level": "INFO"

}

}