

Capstone Project: Building a Machine Learning Model and Developing a Flask Web App

Yuliia Moroz

3013040

Submitted in partial fulfillment for the degree of

B.Sc. in Computing Science

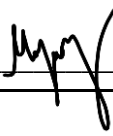
Griffith College Dublin

August, 2024

Under the supervision of Barry Denby

Disclaimer

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the Degree of Bachelor of Science in Computing at Griffith College Dublin, is entirely my own work and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Signed:  _____

Date: 27/08/2024 _____

Acknowledgements

I would like to express my deepest gratitude to my family for being there on my journey. To my grandmother, who was there to support me at my lowest moments; my aunt, who encouraged me not to give up; my mom, who is also trying her best; and my grandfather, who made it possible. You helped me build my strength, brick by brick.

Special thanks to Sofia, my love, who taught me to accept and appreciate myself and Kateryna, for never doubting my abilities. Without your support, I would not have found the motivation to do this for myself.

I am also incredibly grateful to my supervisor, Barry, for his guidance, unwavering support and understanding throughout this project. His patience and encouragement allowed me to work without the fear of criticism, giving me the confidence to reach the end.

Table of Contents

Acknowledgements	iii
List of Figures	v
Abstract	vi
Chapter 1. Introduction	1
Overview	1
Goals and Objectives	1
Overview of Approach	2
Document Structure	3
Chapter 2. Background	4
Literature Review	4
Related Work	5
Chapter 3. Methodology	6
Project Design and Planning	6
Choice of Programming Language	7
Chapter 4. System Design and Specifications	9
Technologies Used	9
System Architecture	11
Process and Data Modelling	12
Challenges and Considerations	12
Chapter 5. Implementation	14
Data Preprocessing and Feature Engineering	14
Loading and Formatting Data	14
Feature Engineering	14
Model Development and Training	15
Linear Regression Model	15
k-Nearest Neighbours (k-NN) Model	16
Long Short-Term Memory (LSTM) Neural Network	16
Integration with Flask Web Interface	17
Flask Application Setup	17
Prediction Function	17
Web Interface	18
Challenges Faced During Implementation	18
Chapter 6. Testing and Evaluation	20
Quantitative Testing	20
Qualitative Testing	22
Evaluation Results	22
Refinements Based on Testing	24
Chapter 7. Conclusions and Future Work	25
Conclusions	25
Future Work	26
References	27

List of Figures

Figure 1 Model Accuracy	21
Figure 2 Model Loss	21
Figure 3 Training and Validation Accuracy	22
Figure 4 Actual vs Predicted Close Prices (LSTM).....	23
Figure 5 Actual vs Predicted Close Prices (k-NN)	23

Abstract

This capstone project focuses on developing and implementing machine learning models to forecast time-series data and creating a Flask web application for user interaction. The project's main goal is the development of the system that can predict the next 5 days' worth of data based on past 30 days information. Three machine learning models were tested and compared: Linear Regression, K-Nearest Neighbours (KNN), and Long Short-Term Memory (LSTM). The LSTM model was finally agreed upon to be used because it had the best performance in capturing the different dependencies of events in time and in prediction of new values. The main purpose of this project as an academic capstone project is to compare different machine learning models, observing how they act under the same conditions and implementing one of the models which fits best to predict and output results using a frontend web application.

This project was chosen as a quick starter to try a new career in data science. By this project, the skills of data analysis, machine learning, and web app development were acquired and the overall idea of the flow of working as a data scientist was given. At the onset of the project, building a stock market predictor was not investigated. However, seeing that stocks and the financial markets have become very important - this project was chosen and became a gateway for real conversations about stocks, investments and financial planning.

The Google_Dataset.csv dataset is one of the components of this project. Through the dataset the first step in the project was to clean the data and make the data more manageable by processing. Data cleaning, feature engineering, and normalization techniques were used to preprocess the data which was later used for model training. In the preprocessing stage, the dataset was then used to train and evaluate the three models. This was accomplished in Python and Jupyter Notebook. The models' performance was verified for different metrics like Mean Absolute Error (MAE) and Mean Squared Error (MSE).

The LSTM model was discovered to be the most accurate, after which it was incorporated into the frontend app made by using Flask. This is a web-based software for the user who offers a specific date, and the computer consequently predicts for the next five days with the data of the previous 30 days. The application predicts only the dates which already exist in the database provided, but it is planned to work on this in the future.

In conclusion, as the system was installed with a tough algorithm and the user should be allowed to communicate very efficiently with it - this project can be considered successful. The effective application of AI and web development not only demonstrates a theoretical concept to solve a real-world problem but also lays a base for other subsequent developments such as using APIs and fine-tuning the LSTM model in order to get more accurate results.

Chapter 1. Introduction

The machine learning of data science, which is a highly dynamic area, understanding and applying advanced technologies is essential for building predictive models that can drive informed decision-making. This capstone project, "Building a Machine Learning Model and Developing a Flask Web App," is meant to make use of artificial intelligence for the time-series forecasting and giving a user-friendly interface for its' users. It demonstrates prediction of five consecutive days of data based on the preceding 30 days, the project proves the viability of machine learning approaches in real-world use cases and provides a look at how these models can be used for the project website through Flask. This project highlights the importance of learning data science, applying new technologies, and independently working on complex tasks.

Overview

The project lies at the cross section of machine learning, time series analysis, and web development. Its prime function is to test a system which can make a prediction of time-series as accurately as possible by employing three distinct machine-learning models: Linear regression, K-Nearest Neighbours (KNN), and Long Short-Term Memory (LSTM). By leveraging historical data from the Google_Dataset.csv file, these models are first trained to predict the future values. The main focus is on the integration of these models into a Python-based environment; Jupyter Notebook provides a platform for model development and a Flask web application is for the deployment.

Respective chosen models represent various ways to approach prediction tasks. Linear Regression, a simple yet powerful statistical method, provides a baseline for comparison. KNN, a non-parametric classification method, handles data by processing the proximity of data points. On the other hand, LSTM, a type of Recurrent Neural Network (RNN), is specifically designed to handle sequential data, making it particularly suitable for time-series forecasting. The project compares these models to determine which one is the best for the dataset and prediction task.

Goals and Objectives

The primary purpose of the research project is to enhance the understanding and skills of data science and machine learning and to come up with a practical application that

showcases the use of these technologies. This project was motivated by the increasing relevance of stock market predictions and financial data analysis. Even though this project focuses only on one stock, it gives a unique chance to learn financial forecasting dynamics and to tackle the problems with accurate trend prediction.

This project aims to:

- To come up with and test three different machine learning models for time-series prediction.
- Find out the most accurate and reliable model for this exact problem.
- Implement a web-based interface by using Flask. This will allow the users to have a conversation with the prediction model, input some dates and get the forecasted data back.
- Use data science and machine learning in real-world scenarios, providing insights in the practice of these concepts.

This project brings out the issue of the importance of knowledge, trying different approaches to find the best solutions and the ability to adapt to technology.

Overview of Approach

The objectives of the project were achieved by using a structured, step-by-step method. The process began with data preparation, involving the cleaning and preprocessing of the Google_Dataset.csv file to assure the quality of the data and its readiness for model training. Next, the development and training of three machine learning models: Linear Regression, KNN, and LSTM using Python in a Jupyter Notebook environment. The model analysis was done with metrics like Mean Absolute Error (MAE) and Mean Squared Error (MSE) to assess the accuracy and the applicability for the forecast of time series.

The LSTM model being the most accurate was, therefore, chosen for the deployment of a web application. As it is a lightweight and flexible web framework, Flask was selected for the purpose of simplicity and ease of integration with Python. Applicability of the Flask-based system lies in the users who feed in a specific date and the system will predict the next five days based on the information of the last 30 days. Through this interactive feature, machine learning's usefulness is realized in a practical way; aside from that, users enjoy an easy interface.

Document Structure

The following chapters make up the rest of the document:

- Chapter Two provides a literature review of the area of time-series forecasting and sources that were consulted to complete the project, in addition to related work. This chapter looks at existing methods and technologies in machine learning and web development.
- In Chapter Three, the requirement analysis and high-level design are explained. It outlines the stages of data preprocessing to model building and the criteria for model validation.
- Chapter Four provides the system design and requirements/specifications, including the tools and software used in the process. It provides a detailed analysis of the models' architectures and the integration of the LSTM model into the Flask web application.
- Chapter Five details the project's implementation. This chapter gives a technological explanation of the coding, model refining, as well as the web app's design.
- Chapter Six explains the functioning of the project through testing and evaluation methods. It also discusses the outcomes regarding any alterations in the overall design and implementation that were required to enhance the performance of the system.
- Chapter Seven is the key points that are the conclusion of the work and future work. It sums up the success of the project, talks about the difficulties encountered, and gives an overview of the potential areas for further development and research.

Chapter 2. Background

This chapter provides an overview of the existing literature and related work connecting to the development of machine learning models for time-series predictions and the development of web apps using Flask. The review encompasses scholarly journals, conference proceedings, books, technical reports, and relevant software documentation to give a deep perception of the issue and to disclose how this project contributes to the already owned knowledge.

Literature Review

Machine Learning Models for Time-Series Forecasting:

Time-series forecasting has been a significant area of research in machine learning, with numerous methodologies developed to predict future values based on historical data. The field has evolved with the introduction of various models, each contributing unique capabilities to handle different aspects of time-series data.

- **Linear Regression Models:** Historically, linear regression was one of the first models to be used in time-series forecasting. Its simplicity and interpretability make it a common baseline model. As per Hyndman and Athanasopoulos (2018), linear regression models are effective for capturing linear trends in data but may struggle with more complex temporal dependencies [1].
- **K-Nearest Neighbours (KNN):** KNN is a non-parametric method that predicts future values based on the similarity of past data points. Research by Fukunaga and Hostetler (1975) and subsequent studies have demonstrated that KNN can capture local patterns in data but may face challenges with scalability and high-dimensional data [2].
- **Long Short-Term Memory (LSTM) Networks:** Long Short-Term Memory (LSTM) networks, which belong to a class of Recurrent Neural Networks (RNN), have been found to be very promising in handling sequential data because they can capture long-term dependencies. Hochreiter and Schmidhuber (1997) and others have shown that LSTM networks are the most efficient and fast tools for time-series forecasting making them the first option in the case of complex temporal and nonlinear patterns [3]. The vanishing gradient problem, which occurs in traditional RNNs, is effectively circumvented by LSTM networks, thus they can produce better performance on long sequences.

A great book about doing data science that was guidance from the start to finish is called “Python Data Science Handbook” by Jake VangerPlas [4]. It touches upon data manipulation, visualization and machine learning.

Web Application Development with Flask:

Flask framework is a very light and easy to use web framework for Python development, so developers generally like it. The design principle of Flask is simplicity and flexibility. It is also very good for developing web applications that interface with machine learning models.

- Flask Framework: Generally, flask's design emphasizes minimalism and extensibility, which makes it the best choice for web applications that will be linked with data processing and machine learning systems. Documentation and guides provided by the Flask project highlight its modular approach, which allows for easy customization and scaling [5].

Related Work

Several projects and research studies have explored aspects related to this project, contributing in one way or another to the development of these time-series forecasting models and web application.

- Great help in achieving splitting for the task at hand was an article called “Simple Sequence Predictions with LSTM” on Medium [6]. It covers an easy to understand approach that was implemented to this capstone project for splitting the dataset to sequences of 30 and 5 days.
- The strongest reference used for the Anaconda model comparison was “Stock Price Prediction Using Multiple Algorithms” project [7]. The functionality is aimed at covering many aspects of data mining which was very useful to learn from.
- There are many models that can be used instead of LSTM. Project called “Stock Price Forecasting Flask Web App” implements an ARMA model with the goal to predict an adjacent close price for the next 7 days [8]. The web application presented there is an inspiration for future work due to its great design and visualization.
- The Stock-Market-AI-GUI project, available on GitHub, is a great example of utilizing machine learning technologies for stock price prediction with an interactive user interface [9]. This project uses deep learning models to predict stock prices and also provides users with a visualization of future price movements based on historical data.

By reviewing the literature and related work, this chapter establishes the context for this project and identifies the advancements made in time-series forecasting and web application development. The following chapters are going to get into methodology, implementation, and results, which will show how this project takes forward and extends the present thinking in these areas.

Chapter 3. Methodology

This chapter describes the approach used to reach the objectives of the capstone project "Building a Machine Learning Model and Developing a Flask Web App." It details technological and design decisions made during the course of the project, thereby giving a reason for these decisions.

Project Design and Planning

The primary goal of the project was to design a system for time series forecasting through machine learning models and to introduce a simple user interface. The design and planning phase included several key decisions regarding the technologies and methodologies used.

1. Choice of Machine Learning Models

Linear Regression, K-Nearest Neighbours (KNN), and Long Short-Term Memory (LSTM) were the three machine learning models examined for time-series forecasting. The Linear Regression model is the simplest one and it was chosen to be a baseline to compare the performance with the added models. The ease of use and the fact that it obtains correct predictions make it an appropriate model for time-series forecasts.

- Linear Regression: it was chosen as a baseline model due to its simplicity and ease of interpretation. It can help in understanding linear relationships in time-series data.
- K-Nearest Neighbours (KNN): The model was chosen for its non-parametric approach. It does not require knowledge of the data distribution. Unlike parametric models, KNN does not assume a specific form for the underlying data distribution, which can be a way to go with the non-linear trends in the time series.
- Long Short-Term Memory (LSTM) Networks: The next model chosen for this project was one which effectively handles long-term dependencies and temporal patterns in sequential data. LSTM's capability handling time-series data and mitigating the vanishing gradient problem made it the most promising model for this project.

2. Data Preparation

This project's dataset came from Kaggle [10], a popular platform for data science competitions and datasets. Data preparation for the 'Google_Dataset.csv' was performed as follows:

- Handling Missing Values: Interpolation methods were used to address missing data points in order to ensure data completeness and consistency.
- Feature Scaling: One of the best methodologies for the model to achieve maximum performance is to normalize or standardize features. This step was

quite vital particularly for models like k-NN which are highly sensitive to the scale of input features.

- Time-Series Formatting: The data has been made into a time-series format suitable for forecasting. That also meant the creation of input features from previous data points and target values for prediction.

Choice of Programming Language

Picking of Python as the project programming language over R was a decision that was made based on several factors, which are:

- Library Support and Ecosystem: Python has a plethora of libraries and built-in frameworks as well as support for the machine learning and web development categories. Libraries like TensorFlow, Keras, and scikit-learn offer in-depth support, while Flask, a web framework, integrates seamlessly with Python, thus leading to flexibility in the developing process.
- Versatility and Integration: Python's flexibility allows it to take a role in different spheres of the project, starting from data preprocessing and model development to web application creation. Besides this, integration makes the project less complex than what it could be by reducing the switching between various kinds of tools and programming languages.
- Community and Resources: Python has a strong and active community, providing countless resources, tutorials, and support. Therefore, a person will easily find solutions to problems they have as well as be in touch with the new trends in data science and web development technology.

4. Model Development and Evaluation

Developing ML models was done in Jupyter Notebook, using Python & its ML libraries for easier handling and integration. The main steps were the following:

- Model Training: Each model was trained using historical data from the dataset. The process of hyperparameter tuning was based on performance indicators obtained through cross-validation.
- Performance Evaluation: The models were tested by calculating Mean Absolute Error (MAE) and Maximum Squared Error (MSE) that show their accuracy. These metrics gave a quantitative measure of the level to which the models made accurate future forecasts.
- Model Comparison: The Linear Regression model, KNN model, and LSTM model were compared to figure out the best one for predicting stock. The main reason the LSTM model was chosen is because it showed the greatest ability to detect the complex temporal patterns in the data.

5. Technology Stack for Web Application

The web application component of this project was created with Flask. In its favour, the following was considered:

- **Simplicity and Flexibility:** Flask's lightweight and modular structure provides simplicity in deployment and flexibility for development. This gave the ability to create a user interface that could talk to the machine learning model without much pain due to it being Python-based together with the machine learning model.
- **Ease of Integration:** Flask comes with an easy solution for connecting Python libraries and tools, thereby making it suitable for the deployment process of machine-learning models. It was a simple solution, which allowed to have a more focused development process.

6. Excluded Technologies and Testing

- **Model Serialization:** Although model serialization was initially planned to save the trained models for future use, it was not implemented during the project. This will be addressed in future work as outlined in Chapter 7.
- **API Development:** The project was done without the inclusion of API development. Primary attention was given to the creation of functional models and their fine-tuning.
- **Testing:** Testing was referred mainly for the purpose of comparing the model's performance using MAE and MSE in the notebook. The web application was manually tested to confirm its functionality. Extending the benefits of automated testing was left aside.

Summary

This chapter highlights the methodology used to develop the machine learning models and the web application for time-series forecasting. The choice of models, data preparation methods, and the use of Python and Flask for development were driven by considerations of simplicity, flexibility, and integration capabilities. The decisions made during the project aimed to balance effectiveness with practical constraints, providing a foundation for future enhancements and deployments. The subsequent chapters will delve into the implementation details, results, and evaluation of the project.

Chapter 4. System Design and Specifications

This chapter gives the insight of the system design and the technologies that are part of the capstone named "Building a Machine Learning Model and Developing a Flask Web App." It describes the technological choices, system architecture, and how various components interact to fulfil the project's objectives. The chapter also discusses the challenges encountered and how the chosen technologies were leveraged to achieve the desired outcomes.

Technologies Used

1. Programming Language: Python

- Version: Python 3.11
- Reason of Choice: Python was chosen because of its extensive use in data science and machine learning. The large number of libraries designed specifically for these tasks made it suitable for handling data analysis, model development, and prediction. Python's simple syntax contributed to ease of learning and development.
- Advantages: Python's extensive ecosystem of libraries for machine learning, data processing, and visualization (such as Pandas, Matplotlib, Scikit-learn, and TensorFlow) significantly enhanced the development process. Its readability improved code maintainability.
- Challenges: As Python is an interpretative coding language it may influence speed of execution. This can occasionally cause problems during the training of computationally demanding models that need a lot of processing power, like the LSTM network.

2. Machine Learning Libraries

- TensorFlow/Keras:
 - - Version: TensorFlow 2.17.0 with Keras API 3.5.0
 - - Reason of Choice: TensorFlow's capability to manage complex neural network architectures and time-series data, with the help of Keras API, made it preferred choice for implementing the LSTM model. GPU acceleration is supported by TensorFlow, which is well optimised for performance and is essential for effectively training deep learning models.
 - - Advantages: TensorFlow's scalability significantly reduced training times for the LSTM model and the high-level interface provided by the Keras API greatly facilitated the process of creating neural networks.
 - - Challenges: TensorFlow's complexity and high learning curve caused initial issues, so careful attention and thorough managing of dependencies were needed to ensure compatibility with other system components.
- Scikit-learn: Used for Linear Regression and KNN models
 - Version: scikit-learn 1.5.1

- Reason of Choice: Scikit-learn was selected because of its reliability and ease of use in implementing traditional machine learning models such as Linear Regression and k-Nearest Neighbours (kNN). Its easily understandable documentation and uniform API allowed for the rapid development and testing of different models.
 - Advantages: The built-in libraries proved to be quite helpful for data preprocessing, model evaluation, and validation. Because of scikit-learn's widespread usage, there is a large community supporting it, which means that materials are easily accessible.
 - Challenges: It is great for standard models, but it doesn't support LSTMs or other complicated deep learning architectures, so it will be required to use a different framework.
- Pandas
 - Version: Pandas 1.3.3
 - Reason of Choice: Pandas is powerful in data manipulation and analysis capabilities. It provides efficient data structures which were essential for handling and preprocessing the time-series data.
 - Advantages: Pandas helped with easy data loading, cleaning, and transformation. Its built-in functions allowed for quick aggregation, filtering, and feature extraction, which were crucial for preparing the data for model training.
 - Challenges: While Pandas is highly effective for data manipulation, it may consume substantial memory for large datasets. Handling very large datasets could lead to performance issues.
 - NumPy
 - Version: Numpy 2.2.2
 - Reason of Choice: Numpy was chosen for its efficiency in numerical computations and array operations. It provided the fundamental tools for working with arrays and performing mathematical operations necessary for data preprocessing and model training.
 - Advantages: Numpy's array operations are optimized for performance and allowed for fast computations. Its support for multi-dimensional arrays and mathematical functions streamlined the data preparation process and enhanced computational efficiency.
 - Challenges: Numpy operations, while fast, require careful management of memory and data types. For very large datasets or complex operations, ensuring efficient use of resources can be challenging and may require additional optimization.

3. Web Framework: Flask

- Vendor/Version: Flask 3.0.3
- Reason of Choice: Ease of use and lightweight nature make it a good choice for creating a simple web application that would interact with the machine learning models. Flask's minimalistic approach adds only the necessary components, reducing operating costs.
- Advantages: Has a seamless way of interaction between the web interface and the backend machine learning models. Rapid development and deployment

were possible by the framework's comprehensive documentation and ease of use.

- **Challenges:** More complicated applications may find Flask restricting due to its lack of built-in support for managing databases and user authentication. These were not necessary for the project's scope.

4. Integrated Development Environment (IDE):

- **Visual Code Studio**
 - **Reason of Choice:** This decision was based on VSC's adaptability, low weight, and robust support for Python development. For this project, its many extensions - for example, version control and Python debugging – made it an especially useful tool.
 - **Advantages:** VS Code's rich extension ecosystem and adaptability facilitated efficient development and debugging. The integration with Git also simplified version control.
 - **Challenges:** VS Code performed well throughout the project, with no significant issues encountered.
- **Data Science Platform: Anaconda**
 - **Reason of Choice:** Anaconda was employed to manage Python packages and environments due to its popularity in data science and its ability to simplify the management of dependencies. The conda package manager made it straightforward to install and maintain the necessary libraries for the project.
 - **Advantages:** Anaconda minimized conflicts by offering a dependable environment for handling Python libraries and dependencies. Anaconda's integration with Jupyter Notebooks facilitated the creation of models and exploratory data analysis.
 - **Challenges:** Anaconda can require a lot of resources, which has occasionally led to slower startup times. Nevertheless, they were reasonable and had no negative effects on the project.

System Architecture

The system architecture is supposed to pair the artificial intelligence models with a web application, setting the conditions for the user to input their data and get the forecasts. The architecture is built on a tiered approach where the data is processed, the models are trained, and a web interface is developed.

1. Architecture Diagram

The system architecture includes three main tiers:

- **Data Processing Tier:** This tier is responsible of taking care of the data cleaning, preprocessing, and transformation tasks. It acts as a data preparation for model training and evaluation.
- **Machine Learning Tier:** It is devoted to the implementation and training of machine learning models. This tier consists of Linear Regression, KNN, and

LSTM models. The LSTM model was selected based on performance evaluations.

- Web Application Tier: Handles user interactions and provides predictions. Developed with Flask, the tier interacts with the frontend by receiving and processing the request, works with the LSTM model, and displays the outcomes to the user.

2. Integration and Data Flow

- Data Flow: The flow of data starts from the dataset that is pre-processed in the Data Processing Tier. The cleaned and formatted data is what comes next and is used to train the machine learning models in the Machine Learning Tier. This then is incorporated into the Web Application Tier, where users input dates to receive the predictions.
- Interaction: The Flask web interface is what is responsible for connecting to the LSTM model, which gives the forecasts. The user input undergoes transformation and is directed to the model, which computes the prediction of the given date.

Process and Data Modelling

Process Modelling

Database management system lifecycle:

- Data Preparation: Cleaning and transforming data into a suitable format for machine learning.
- Splitting: Splitting the data for prediction.
- Model Training: Teaching different models using historical data.
- Predicting: Doing the predictions.
- Evaluation: Assessing model performance using metrics like MAE and MSE.
- Web Interface Development: Developing a user interface which can operate the prepared model.

Challenges and Considerations

Difficulties Encountered

- Model Training: Training the LSTM model required substantial computational resources and time to tune hyperparameters effectively.
- Web Development: Integrating the machine learning model with the Flask web application involved handling data interchange and ensuring smooth interaction between components. Some problems occurred during this stage of the project but were quickly mitigated by focusing on data types used.

Summary

This chapter provides an in-depth look at the technologies used, the system architecture, and the design considerations for the capstone project. It describes the integration of machine learning models with a web application and highlights the challenges and solutions encountered during the development process. The chapters that follow will discuss the implementation details, including code snippets and specific technical aspects of the project.

Chapter 5. Implementation

This chapter outlines the of the stock price prediction system, outlining the steps taken to create a working version of the system. The whole project was a combination of AI techniques and their integration in the web-based app to predict stock prices. The implementation was divided into three primary parts: data preprocessing and feature engineering, model development and training, and integration of the prediction model with a Flask-based web interface. This chapter will cover each of these components and explain in detail steps taken.

Data Preprocessing and Feature Engineering

This stage of implementation focuses on preparing the data for model training. The data used was a dataset of Google's stock prices, stored in a CSV file named 'Google_dataset.csv'. It contains daily records of the opening, closing, high, low prices, and volume of trades.

Loading and Formatting Data

After loading the dataset to Pandas DataFrame, the 'Date' column is converted to a datetime format and set as the index to facilitate easy time series manipulation. This way the data can be now easily accessed and manipulated by date, which will be essential for time series forecasting. During this time, it is also ensured that the dataset does not hold null values or missing values, and if all the values are the right datatypes.

Feature Engineering

Afterwards, the data was split into features and targets using a sliding window approach to capture temporal dependencies. The `create_features_and_target` function was developed to generate the required features and targets:

```
def create_features_and_target(data, past_days, future_days):
    X, y = [], []
    for i in range(len(data) - past_days - future_days + 1):
        X.append(data.iloc[i:i + past_days].values.flatten())
        y.append(data.iloc[i + past_days:i + past_days +
future_days].values.flatten())
    return np.array(X), np.array(y)

#set past and future days
past_days = 30
future_days = 5
```

```
#creating new datasets  
dataClose = df[['Close']]
```

```
#create features and targets  
X, y = create_features_and_target(dataClose, past_days, future_days)
```

This means that each feature vector comprised the closing prices of the past 30 days, and the corresponding target vector consisted of the closing prices of the next 5 days.

Model Development and Training

The next phase after preprocessing is all about developing and training different machine-learning models. Three models were considered: Linear Regression, k-Nearest Neighbours (k-NN), and a Long Short-Term Memory (LSTM) neural network. Each model was trained and evaluated separately to identify the best-performing approach.

Linear Regression Model

The Linear Regression model was used as a baseline model due to its simplicity and interpretability. The dataset was split into training and testing sets, scaled, and then fed into the Linear Regression model for training:

```
#create features and targets  
X, y = create_features_and_target(dataClose, past_days, future_days)
```

```
#split the data into training and testing sets  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
shuffle=False)
```

```
#scale data  
scaler_X = StandardScaler()  
X_train = scaler_X.fit_transform(X_train)  
X_test = scaler_X.transform(X_test)
```

```
#scale target data  
scaler_y = StandardScaler()  
y_train = scaler_y.fit_transform(y_train)  
y_test = scaler_y.transform(y_test)
```

```
#initialize the Linear Regression model  
lr_model = LinearRegression()
```

```
#train the model  
lr_model.fit(X_train, y_train)
```

```
#make predictions  
LR_pred = lr_model.predict(X_test)
```

The model's predictions were inverse-transformed to match the original scale of the 'Close' prices, and the performance was evaluated using Mean Squared Error (MSE).

k-Nearest Neighbours (k-NN) Model

The k-NN model was explored as an alternative to the Linear Regression model. The GridSearchCV method was utilized to find the optimal number of neighbours:

```
#initialize KNeighborsRegressor  
knn = KNeighborsRegressor()  
  
#define the parameter grid  
knn = KNeighborsRegressor()  
params = {'n_neighbors': range(1, 21)}  
knn_model = GridSearchCV(knn, params, cv=5)  
knn_model.fit(X_train, y_train)  
  
#make predictions  
KNN_pred = knn_model.predict(X_test)
```

Long Short-Term Memory (LSTM) Neural Network

The LSTM model was chosen for its ability to capture long-term dependencies in time series data. The input data was reshaped to fit the expected input shape of the LSTM:

```
#reshape for LSTM input (samples, timesteps, features)  
X = X.reshape(X.shape[0], X.shape[1], 1)  
y = y.reshape(y.shape[0], y.shape[1])  
  
#define the LSTM model  
model = Sequential()  
model.add(Input(shape=(X_train.shape[1], 1)))  
model.add(LSTM(units=50, return_sequences=True))  
model.add(Dropout(0.1))  
model.add(LSTM(units=50))  
model.add(Dropout(0.1))  
model.add(Dense(units=forecast_length))  
#compile the model
```

```
model.compile(optimizer=Adam(learning_rate=0.001),
loss='mean_squared_error', metrics=['mae', 'accuracy'])
```

train the model

```
history = model.fit(X_train, y_train, epochs=11, batch_size=32, verbose=1,
validation_split=0.2)
```

#make predictions with the LSTM model

```
predictions = model.predict(X_test)
```

The LSTM model was trained over multiple epochs, and the performance was evaluated using the same metrics as the other models.

Integration with Flask Web Interface

The final phase of implementation involved the development of a Flask-based web application using LSTM. With this users now can observe the results of their predictions using a simple, but interactive interface where they can input a target date and then receive prediction of stock prices for the following five days.

Flask Application Setup

A Flask application was created to handle user inputs, make predictions using an LSTM model, and render results on a web page. The application's main file, app.py, defined routes for the home page and handled form submissions.

Prediction Function

The predict_next_5_days function was defined in modelLSTM.py to generate predictions based on the user-provided target date. The function validates the date, checks if sufficient data is available, and uses the model to predict the next five days' stock prices:

```
def predict_next_5_days(data, target_date):
    target_date = pd.to_datetime(target_date, format='%d/%m/%Y')

    if target_date.weekday() >= 5:
        return None, "Target date falls on a weekend. Please choose a weekday."

    if target_date not in df.index:
        return None, f"Target date {target_date.date()} not found in the data.
Please provide a valid date."
```

```

target_index = df.index.get_loc(target_date)

if target_index < 30 or target_index + 5 > len(df):
    return None, "Not enough data to predict for this date."

#prepare the input for prediction
last_sequence = df['Close'].values[target_index -
30:target_index].reshape(1, 30, 1)
#predict the next 5 days
predictions = model.predict(last_sequence, verbose=0)

return predictions, None

```

Web Interface

The web interface was designed using HTML and Jinja templating in the index.html file. Users could input a target date, and the interface would display the predicted stock prices, or any errors encountered during prediction.

Challenges Faced During Implementation

During the implementation process several challenges were encountered:

- **Overfitting in models:** One of the primary challenges was dealing with overfitting, especially with the LSTM and k-NN models. While the LSTM model was doing great on training data, the performance decreased when working with testing data. This can indicate that the model was learning too much from the training data, including noise, making it less effective when presented with new data. Regarding k-NN it also faced overfitting issues, struggling to generalize well due to the large size of the dataset and its sensitivity to local data patterns.
- **Fine-tuning LSTM Layers:** Implementing the LSTM model presented its own set of challenges, particularly in determining the optimal architecture and parameters. It was quite a challenge to balance the number of layers and units within layers so that the model could learn effectively without overfitting. Too many led to overfitting, while too few resulted in underfitting. Models that had many layers and units overfitted the data, while those that had few layers and units underfitted the data. Several settings had to be changed, such as the number of LSTM layers, the number of units per layer, and the dropout rates. And the process was very time-consuming because of the fact that every change would require extensive training time. It was by trying out different configurations and testing that a tolerable blend between complexity and adequacy was found for now.
- **Ensuring Consistent Predictions:** Achieving reliable predictions from the LSTM model also required careful handling of input data shapes and scaling.

Data had to be converted to a three-dimensional format suitable for the LSTM model, which also included an effective manipulation of NumPy arrays. The feature and target data were scaled appropriately to ensure consistency during training and testing. Ensuring that these data preparation steps were handled correctly was crucial for the model's performance and avoiding prediction errors.

- **Model Serialization:** A difficulty that came up was the task of saving and loading the LSTM model so that the Flask application could use it efficiently. This issue was investigated, but not implemented. With the use of Keras' `save` and `load_model` functionalities, the model would be able to quickly load into memory when the application starts.

Summary

Chapter 5 has outlined the detailed steps taken to implement the stock price prediction system, from data preprocessing to integrating the model into a web application. Each part of the project has been described with accompanying code snippets. It was characteristic of the project to be precise and to take one problem at a time. By employing a systematic approach, the project successfully created a functional web-based stock price prediction tool that allows users to forecast Google's stock prices for chosen 5 days.

Chapter 6. Testing and Evaluation

Testing and evaluation are critical components in the development process, providing insight into the system's performance and identifying areas of improvement. This chapter explains the testing and evaluation methods and the steps that are implemented to present the level of effectiveness and dependability of the stock price prediction system.

Quantitative Testing

Quantitative testing primarily involved evaluating the predictive accuracy of the ML models using standard metrics. These are the steps that were taken to assess the models:

- **Evaluation Metrics:**
The Mean Squared Error (MSE) and Mean Absolute Error (MAE) metrics were used to measure the performance of the prediction models. These metrics provide insights into the average magnitude of errors made by the models, with lower values indicating better performance.
- **Model Training and Testing:**
The data was split into training and testing sets, where 80% was used for training and 20% for testing. This split helped test the models' capability to generalize new forecasted data.
- **Model Evaluation:**
Each model was trained and tested with the prepared datasets and analysed for performance using MSE.
- **Moreover,** the number of neighbours for k-NN was chosen by GridSearchCV, and the LSTM model was implemented and trained by the TensorFlow system, using several LSTM layers and dropout for regularization.
- **Results Visualization:**
Graphs were created to plot the relationship between predicted and actual stock prices, model accuracy, model loss and training and validation accuracy to visualize the efficiency of the developed models.

Model Accuracy:

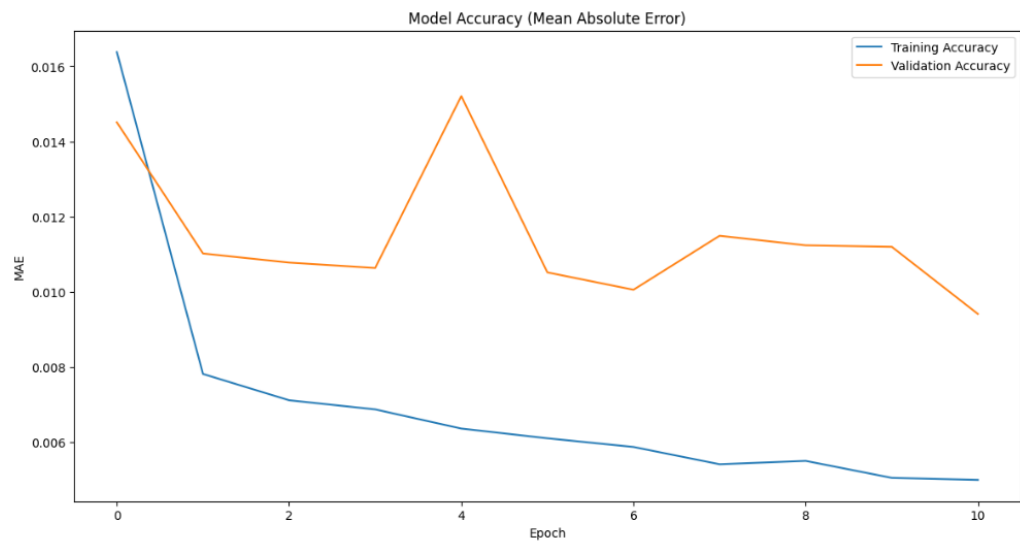


Figure 1 Model Accuracy

Model Loss:

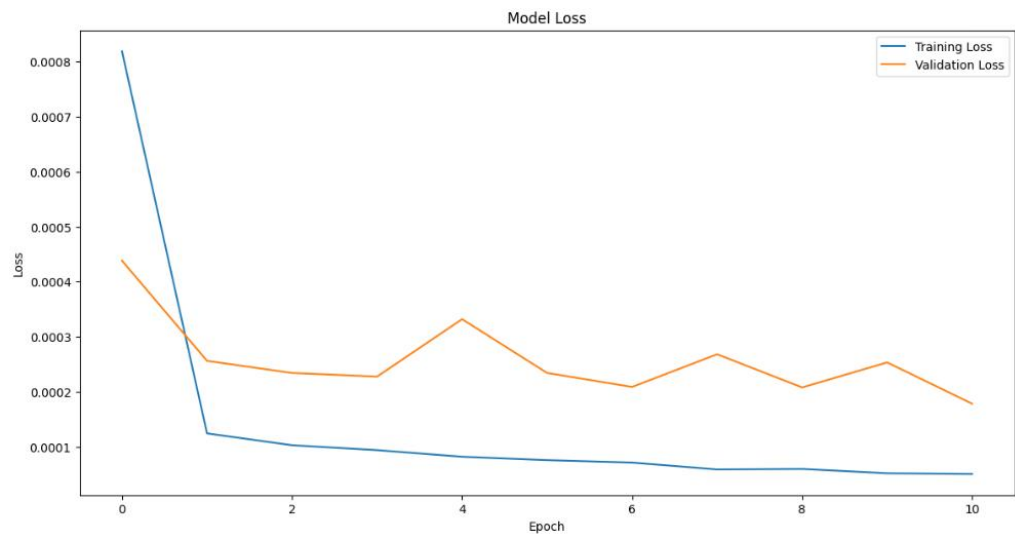


Figure 2 Model Loss

Training and Validation Accuracy:

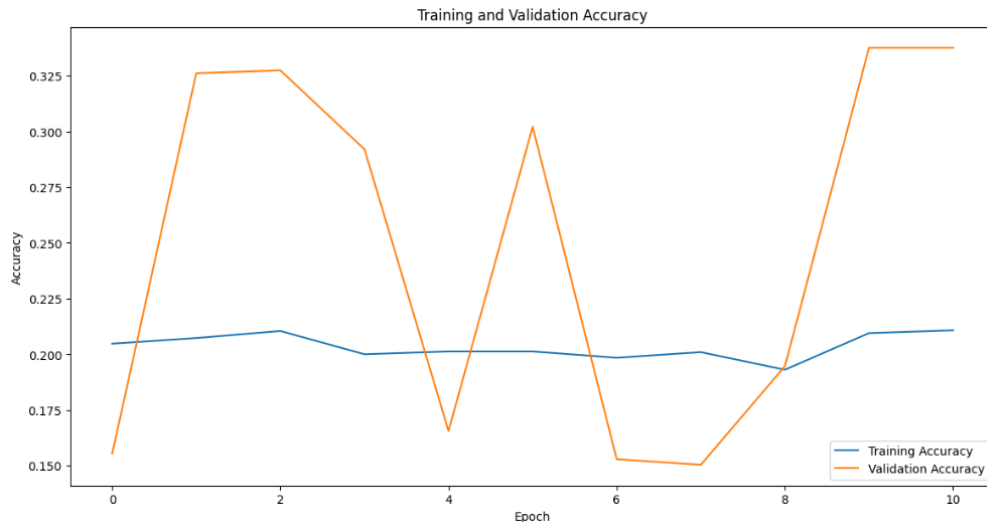


Figure 3 Training and Validation Accuracy

Qualitative Testing

The qualitative testing focused on the usability and functionality of the Flask-based web application. The following method was employed:

- **Manual Testing:** The web application was manually tested to ensure that all features worked as expected. This meant entering valid and invalid dates, checking error handling mechanisms, and verifying the correctness of the predicted outputs. Special attention was given to edge cases, such as selecting dates on weekends.

Evaluation Results

The evaluation and testing procedure provided insights into the performance of the system, are as follows:

- **Model Accuracy:** The LSTM model demonstrated superior accuracy compared to Linear Regression and k-NN models. The LSTM's ability to capture complex temporal patterns made it well-suited for stock price prediction, which reflected in its lower MSE and MAE values. However, it still has some issues with overfitting.

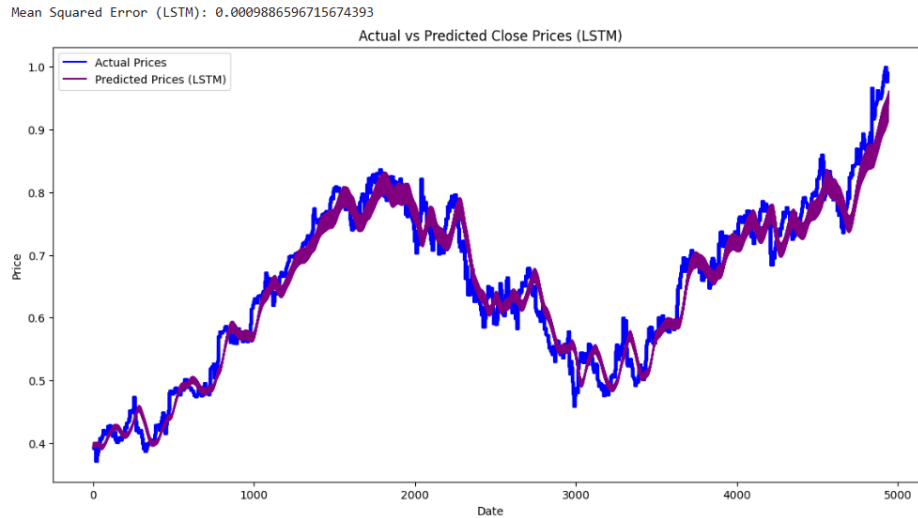


Figure 4 Actual vs Predicted Close Prices (LSTM)

On the other hand, the k-NN model performed the worst among the evaluated models, primarily due to the large size of the dataset. K-NN's effectiveness typically relies on finding local data patterns, but with an extensive dataset like the Google stock prices from 1970 to 2024, the model struggled to generalize well. This posed an issue that led to increased computational demands and resulted in poor predictive accuracy compared to the Linear Regression and LSTM models.

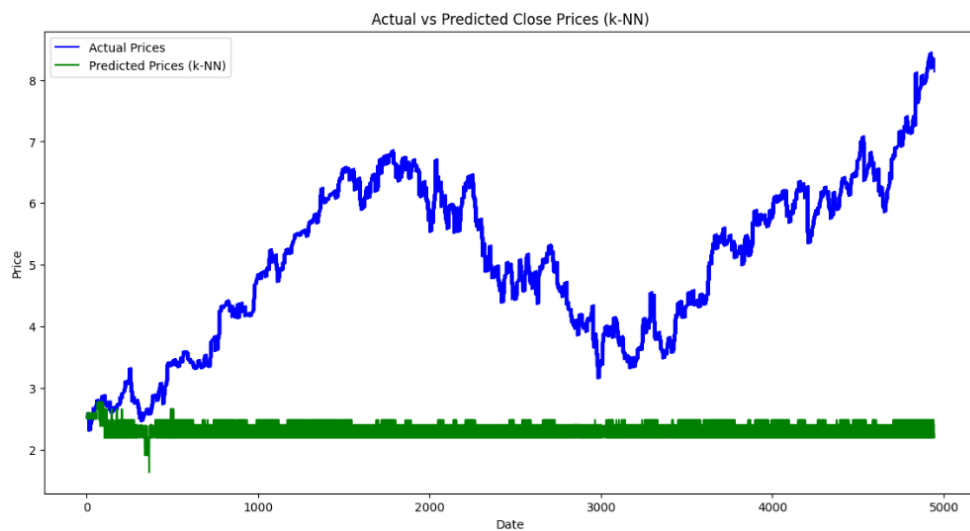


Figure 5 Actual vs Predicted Close Prices (k-NN)

- **Overfitting Management:** Overfitting was one of the issues discovered during initial testing sessions, especially with the two models--k-NN and LSTM. The effects of the strategies were significant as cross-validation brought in and, a dropout approach was used in the LSTM architecture which helped to overcome the overfitting problem and, thus, to enhance their adaptability to new data.

Refinements Based on Testing

Based on the evaluation metrics, adjustments were made to the LSTM model's architecture, including the number of LSTM layers, the number of units per layer, and the learning rate. These adjustments aimed to improve the model's predictive accuracy while reducing overfitting.

Summary

In conclusion, the testing and evaluation process provided valuable insights into the strengths and limitations of the stock price prediction system. The use of quantitative and qualitative testing methods helped ensure that the system was more or less accurate, reliable, and user-friendly. Despite encountering some challenges, such as overfitting and model fine-tuning, the project achieved its objectives, delivering a functional web application capable of making accurate stock price predictions. Future work will focus on further improving model accuracy, scalability, and user experience.

Chapter 7. Conclusions and Future Work

Conclusions

This project aimed to develop and evaluate a stock price prediction system using various machine learning models, such as Linear Regression, k-Nearest Neighbours (k-NN), and Long Short-Term Memory (LSTM) networks. The objectives were to create three forecast models, compare and choose the one most fitting for stock forecasting, and then produce an easily usable web application, respectively.

Achievements:

- **Model Development: Linear Regression:** This baseline model was the first one that entered the project. Although it indeed has a respectable accuracy with a mean square error (MSE) of 14.849461, it was not very effective as other more complex models were.
- **k-Nearest Neighbours (k-NN):** The large dataset which extended from year 1970 to 2024 made k-NN too computationally intensive, thereby less efficient. The model had problems with scaling and a higher MSE than both Linear Regression and LSTM, highlighting its limitations in handling large datasets and capturing temporal patterns.
- **LSTM Network:** The LSTM model emerged as the most successful, with the lowest MSE of 0,00098865. Its power model sequential dependencies made it a very successful time series forecasting tool.

Web Application:

- Developed a Flask-based web application that allows users to interact with the LSTM prediction model. The application provides an intuitive interface for users to input dates and receive predictions. It effectively handled various inputs and provided clear error messages.

Limitations:

- **Model Overfitting:** Although the LSTM model still had highly accurate results, it showed an overfitting problem. There were several regularization techniques applied that helped finding the balance the performance, but the problem persists. Prediction does not suffer from this gravely and does well, but this may become a problem when implementing less limited stock prediction that will perform with generalizing data.
- **Data Constraints:** The inability to predict on new unseen data.
- **Simplicity:** Web Application suffers from this the most as there is not much functionality. Possible solutions to this will be discussed next in future work.

Future Work

Enhancements and Next Steps:

- **Handling Overfitting:** The implementation of more advanced quality regularization techniques together with extensive hyperparameter tuning to refine overfitting, in particular, for the LSTM model.
- **Cross-Validation:** Using robust cross-validation methods to better assess model performance and generalizability.
- **User Experience Enhancements:** The user interface is planned to be enhanced, which includes the features such as historical data visualization and interactive charts that users can utilize to get more profound insights.
- **API Development:** Construct APIs for seamless integration with other applications and services, enabling more versatile use of the prediction models.
- **Cloud Deployment:** Explore cloud-based deployment options to improve accessibility and scalability of the web application and models.
- **Model Serialization:** With the use of Keras' save and load_model functionalities, the model could be able to quickly load into memory when the application starts.

In summary, even though the project successfully achieved its main objectives, the outlined future work will address current limitations and explore opportunities for further development, enhancing the prediction system's accuracy, usability, and scalability.

References

- [1] R. J. H. a. G. Athanasopoulos, "Forecasting: Principles and Practice," Otexts, 2018. [Online]. Available: <https://otexts.com/fpp2/index.html>.
- [2] K. F. a. L. Hostetler, The estimation of the gradient of a density function, with applications in pattern recognition, 1975.
- [3] S. & S. Hochreiter, "Long short-term memory," *Neural Computation* 9(8), 1997.
- [4] J. VanderPlas, "Python Data Science Handbook," [Online]. Available: <https://jakevdp.github.io/PythonDataScienceHandbook/>. [Accessed 05 2024].
- [5] "Flask," Pallets, 2010. [Online]. Available: <https://flask.palletsprojects.com/en/3.0.x/>. [Accessed 07 2024].
- [6] Nutan, "Simple Sequence Prediction With LSTM," 12 03 2021. [Online]. Available: <https://medium.com/@nutanbhogendrasharma/simple-sequence-prediction-with-lstm-69ff0f4d57cd>. [Accessed 07 2024].
- [7] A. Louly, "Stock Price Prediction Using Multiple Algorithms," [Online]. Available: <https://github.com/AdamLouly/Stockprediction>. [Accessed 15 08 2024].
- [8] A. Iwunze, "Stock Price Forecasting Flask Web App," [Online]. Available: <https://github.com/arithescientist/spf>. [Accessed 04 07 2024].
- [9] A. S. Hussain, "Stock-Market-AI-GUI," 18 08 2020. [Online]. Available: <https://github.com/crypto-code/Stock-Market-AI-GUI>.
- [10] M. Anand, "Google Stock Price Dataset," [Online]. Available: <https://www.kaggle.com/datasets/mayankanand2701/google-stock-price-dataset?resource=download>. [Accessed 05 2024].