Assignment-I

Stock Web Oracle

Web Development Framework Using Flask (23AI002)

Submitted by

Shail Sharma

2210993837

Semester: 4

BE-CSE (Artificial Intelligence)

Submitted To

Dr. Rajan Kumar

Assistant Professor, Department of CSE(AI),

CUIET, Chitkara University

CHITKARA UNIVERSITY INSITUTE OF ENGINEERING & TECHNOLOGY

CHITKARA UNIVERSITY, RAJPURA

March, 2024

Table of Contents

1. Introduction

In the fast-paced and unpredictable world of financial markets, the ability to predict stock prices accurately has long been a coveted skill. With the advent of machine learning (ML) techniques, there's been a surge in leveraging data-driven approaches to forecast stock price movements. This project endeavours to harness the power of ML algorithms within a user-friendly web application developed using Flask, a Python web framework. By amalgamating historical stock data, sophisticated predictive models, and intuitive web interface design, this project aims to offer users a convenient platform for making informed decisions in the volatile landscape of stock trading.

The primary goal of this project is to develop a robust system that provides accurate predictions of future stock prices based on historical data and relevant market features. The integration of machine learning algorithms facilitates the extraction of intricate patterns and relationships within the data, enabling the creation of predictive models capable of discerning trends and making forecasts. Through the utilization of Flask, an agile and scalable web framework, these predictive capabilities are made accessible to users through an intuitive and interactive interface.

By combining the prowess of machine learning with the accessibility of web development, this project seeks to democratize stock price prediction, empowering investors, traders, and financial enthusiasts with actionable insights. The user-friendly nature of the Flask web application ensures that even individuals with limited technical expertise can navigate the complexities of stock market predictions with ease. Furthermore, the iterative nature of the project allows for continuous refinement and improvement based on user feedback and evolving market conditions.

In summary, this project represents a convergence of cutting-edge technology and practical utility, bridging the gap between advanced predictive modelling and user-centric design. Through the synthesis of machine learning algorithms, Flask web development, and a user-friendly interface, this endeavour aims to democratize access to stock price predictions, empowering individuals to navigate the intricacies of financial markets with confidence and informed decision-making.

2. Methodology

1. Data Collection and Preprocessing:

- ✓ Collect historical stock data from reliable financial sources.
- ✓ Preprocess the data by handling missing values, scaling, and transforming features as needed.
- ✓ Engineer relevant features that might contribute to improved model performance.

2. Machine Learning Model:

- ✓ Explore and choose appropriate machine learning algorithms for time series prediction, such as LSTM (Long Short-Term Memory) networks, ARIMA (Auto Regressive Integrated Moving Average), or ensemble methods.
- ✓ Train the model using historical stock data, optimizing hyperparameters for better accuracy.
- ✓ Evaluate the model's performance using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and accuracy.

3. Flask Web Application:

- ✓ Develop a web application using Flask, a lightweight web framework in Python.
- ✓ Create user interfaces for inputting stock data, selecting features, and specifying the prediction timeframe.
- ✓ Integrate the trained machine learning model into the Flask application.

4. User Authentication and Security:

- ✓ Implement user authentication to ensure that only authorized users can access the prediction features.
- ✓ Secure the application by validating inputs and implementing secure coding practices.

5. Deployment:

- ✓ Deploy the Flask application on a web server to make it accessible to users.
- ✓ Ensure scalability and responsiveness for a seamless user experience.

3. Tools and Technologies

1. Python:

Python serves as the primary programming language for this project due to its versatility, ease of use, and extensive libraries for data analysis, machine learning, and web development.

2. Flask:

Flask is utilized as the web framework for developing the user interface and backend functionality of the web application. Its lightweight nature and flexibility make it well-suited for building scalable web applications.

3. Pandas and NumPy:

Pandas and NumPy are essential libraries in Python for data manipulation and numerical computing, respectively. They are used extensively for data preprocessing, feature engineering, and model training.

4. Scikit-learn:

Scikit-learn provides a wide range of machine learning algorithms and tools for model selection, training, and evaluation. It is employed for building and training predictive models based on historical stock data.

5. HTML/CSS/JavaScript:

HTML, CSS, and JavaScript are used for frontend development to design the user interface and create interactive elements within the web application. They ensure a visually appealing and intuitive user experience.

6. SQL Alchemy or MongoDB:

SQL Alchemy, an Object-Relational Mapping (ORM) library, or MongoDB, a NoSQL database, may be used for managing and persisting user data, historical stock data, and model configurations within the application.

7. Git/GitHub:

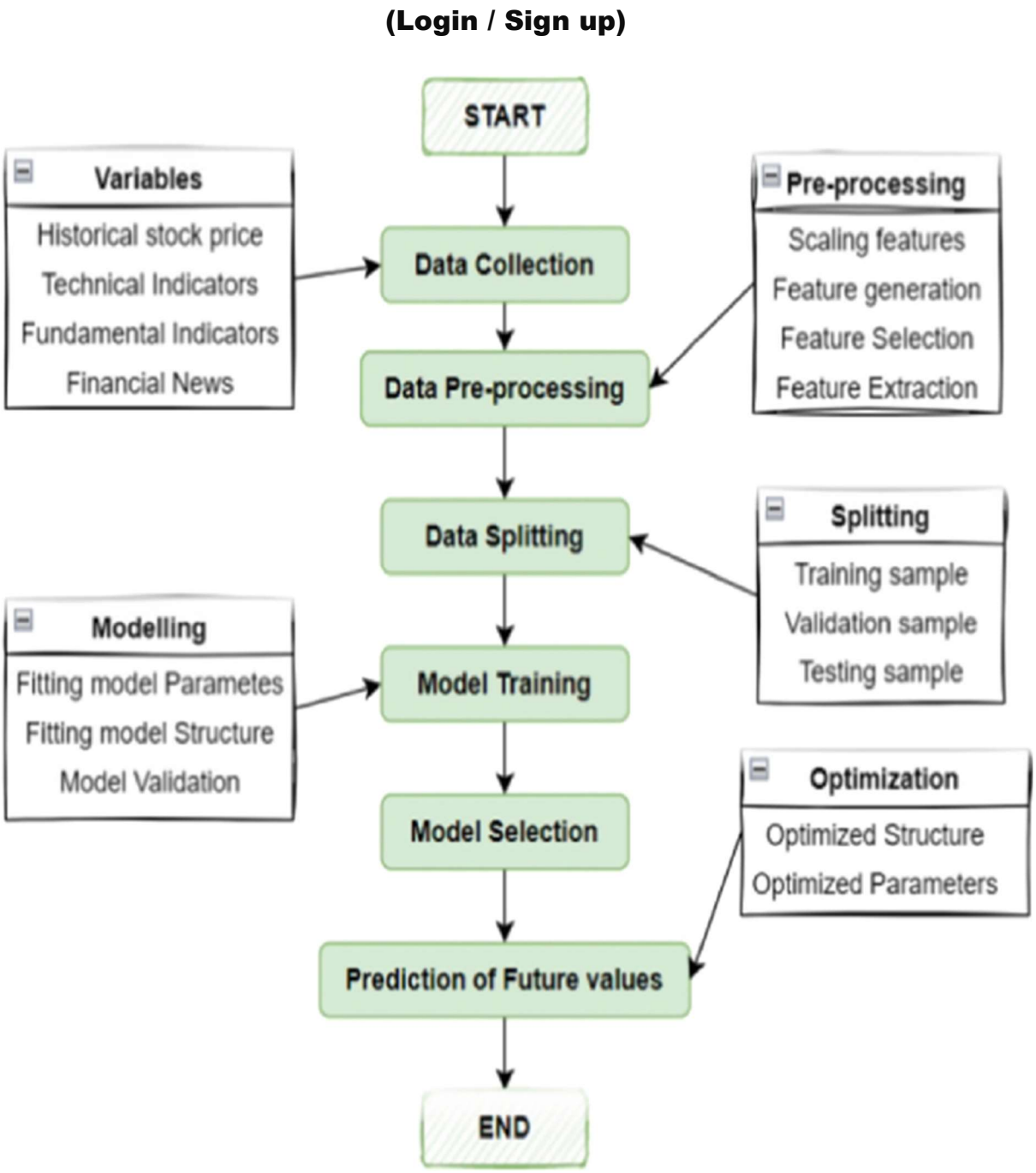
Version control systems like Git and platforms like GitHub are employed for collaborative development, tracking changes, and ensuring code integrity throughout the project lifecycle.

8. VS Code:

In this project, Visual Studio Code (VS Code) streamlines code editing, debugging, and version control tasks, enhancing developer productivity and facilitating collaborative development. Its extensive extensions ecosystem and task automation capabilities

further optimize the development workflow for efficient implementation of machine learning models and Flask web applications.

4. Flowchart



5. Code (Python File Only)

```
   # Importing flask module in the project is mandatory
# An object of Flask class is our WSGI application.
from flask import Flask, render_template, request, send_from_directory
import utils
import train_models as tm
import os
import pandas as pd

# Flask constructor takes the name of
# current module (__name__) as argument.activate
app = Flask(__name__)
#
# @app.route('/favicon.ico')
# def favicon():
#     return send_from_directory(os.path.join(app.root_path, 'static'),
#                                'favicon.ico',mimetype='image/vnd.microsoft.icon')


def perform_training(stock_name, df, models_list):
    all_colors = {'SVR_linear': '#FF9EDD',
                  'SVR_poly': '#FFFD7F',
                  'SVR_rbf': '#FFA646',
                  'linear_regression': '#CC2A1E',
                  'random_forests': '#8F0099',
                  'KNN': '#CCAB43',
                  'elastic_net': '#CFAC43',
                  'DT': '#85CC43',
                  'LSTM_model': '#CC7674'}

    print(df.head())
    dates, prices, ml_models_outputs, prediction_date, test_price =
tm.train_predict_plot(stock_name, df, models_list)
    origdates = dates
    if len(dates) > 20:
        dates = dates[-20:]
        prices = prices[-20:]

    all_data = []
    all_data.append((prices, 'false', 'Data', '#000000'))
    for model_output in ml_models_outputs:
        if len(origdates) > 20:
            all_data.append(
```

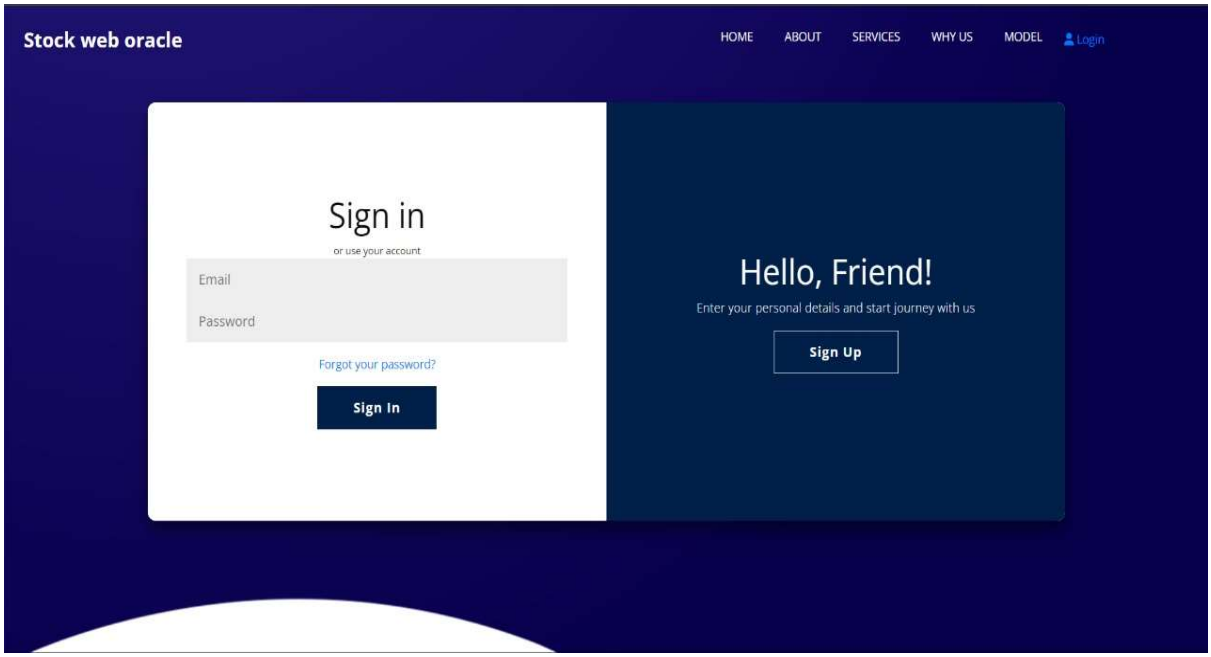
```
                (((ml_models_outputs[model_output])[0])[-20:], "true",
model_output, all_colors[model_output]))
        else:
            all_data.append(
                (((ml_models_outputs[model_output])[0]), "true", model_output,
all_colors[model_output]))

    all_prediction_data = []
    all_test_evaluations = []
    all_prediction_data.append(("Original", test_price))
    for model_output in ml_models_outputs:
        all_prediction_data.append((model_output,
(ml_models_outputs[model_output])[1]))
        all_test_evaluations.append((model_output,
(ml_models_outputs[model_output])[2]))

    return all_prediction_data, all_prediction_data, prediction_date, dates,
all_data, all_data, all_test_evaluations

all_files = utils.read_all_stock_files('individual_stocks_5yr')
# The route() function of the Flask class is a decorator,
# which tells the application which URL should call
# the associated function.
@app.route('/')
# ‘/’ URL is bound with hello_world() function.
def landing_function():
    # all_files = utils.read_all_stock_files('individual_stocks_5yr')
    # df = all_files['A']
    # # df = pd.read_csv('GOOG_30_days.csv')
    # all_prediction_data, all_prediction_data, prediction_date, dates,
all_data, all_data = perform_training('A', df, ['SVR_linear'])
    stock_files = list(all_files.keys())

    return render_template('index.html',show_results="false",
stocklen=len(stock_files), stock_files=stock_files, len2=len([]),
                           all_prediction_data=[],
                           prediction_date="", dates=[], all_data=[],
len=len([]),all_files=all_files)

@app.route('/process', methods=['POST'])
def process():

    stock_file_name = request.form['stockfile']
    ml_algoritms = request.form.getlist('mlalgos')

    # all_files = utils.read_all_stock_files('individual_stocks_5yr')
    df = all_files[str(stock_file_name)]
    # df = pd.read_csv('GOOG_30_days.csv')
```

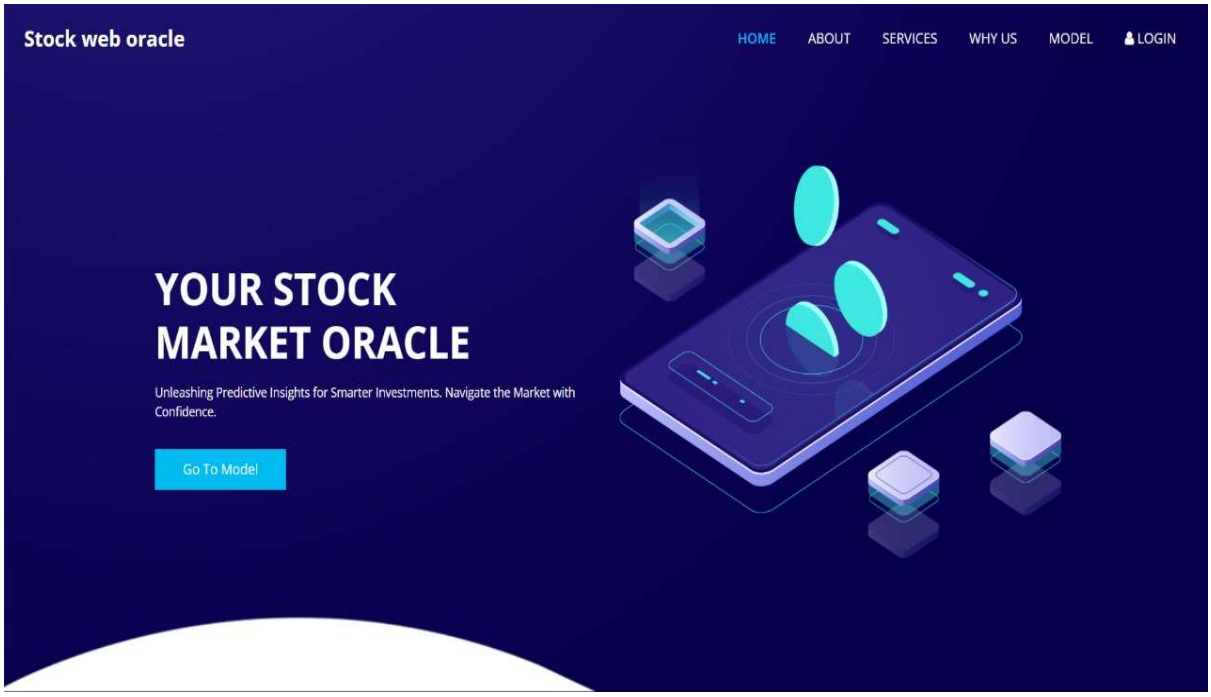
```
    all_prediction_data, all_prediction_data, prediction_date, dates,
all_data, all_data, all_test_evaluations =
perform_training(str(stock_file_name), df, ml_algoritms)
    stock_files = list(all_files.keys())

    return
render_template('index.html',all_test_evaluations=all_test_evaluations,
show_results="true", stocklen=len(stock_files), stock_files=stock_files,
                            len2=len(all_prediction_data),
                            all_prediction_data=all_prediction_data,
                            prediction_date=prediction_date, dates=dates,
all_data=all_data, len=len(all_data), all_files=all_files)

# main driver function
if __name__ == '__main__':
    # run() method of Flask class runs the application
    # on the local development server.
    app.run(debug=True)
```

```
import os
import pandas as pd

def read_all_stock_files(folder_path):
    allFiles = []
    for (_, _, files) in os.walk(folder_path):
        allFiles.extend(files)
        break

    dataframe_dict = {}
    for stock_file in allFiles:
        df = pd.read_csv(folder_path + "/" +stock_file)
        dataframe_dict[(stock_file.split('_'))[0]] = df
    return dataframe_dict

ans = read_all_stock_files('individual_stocks_5yr')
print(ans)
```

```
# Import the libraries
import pandas as pd
import matplotlib.pyplot as plt
import utils
import numpy


# Load the data
# from google.colab import files # Use to load data on Google Colab
# uploaded = files.upload() # Use to load data on Google Colab

def create_plot(dates, original_prices, ml_models_outputs):
    plt.scatter(dates, original_prices, color='black', label='Data')
    for model in ml_models_outputs.keys():
        plt.plot(dates, (ml_models_outputs[model])[0],
color=numpy.random.rand(3, ), label=model)

    plt.xlabel('Days')
    plt.ylabel('Price')
    plt.title('Regression')
    plt.legend()
    plt.savefig("Plot.png")
    plt.show()


def train_predict_plot(file_name, df, ml_model):

    # print (df.head())

    ml_models_outputs = {}

    dates, prices, test_date, test_price = utils.getData(df)
    # utils.LSTM_model(dates, prices, test_date, df)
    for model in ml_model:
        method_to_call = getattr(utils, model)
        ml_models_outputs[model] = method_to_call(dates, prices, test_date,
df)

    dates = list(df['date'])
    predict_date = dates[-1]
    dates = dates[:-3]
    # create_plot(dates, prices, ml_models_outputs)
    return dates, prices, ml_models_outputs, predict_date, test_price

# train_predict_plot('GOOG_30_days.csv', ['LSTM_model', 'elastic_net', 'BR'])

# print (all_files.keys())
```

6. Major Findings/Outcomes/Output





About Us

Stock web oracle is a pioneering platform harnessing AI to deliver accurate stock predictions and empower investors with actionable insights.

We Are Stock web oracle

Stock web oracle is revolutionizing the way investors navigate the stock market. With a focus on innovation and accuracy, our platform leverages cutting-edge AI technology to provide users with predictive analytics and personalized recommendations. Whether you're a seasoned investor or just starting out, Stock web oracle equips you with the tools and insights needed to make informed decisions and stay ahead of market trends.

Join us on this journey to unlock the potential of your investments

Our Services

Your Smart Companion for Predictive Stock Insights

ADVANCED PREDICTIVE ALGORITHMS

Employ cutting-edge machine learning techniques to analyze historical stock data and predict future trends with high accuracy

USER-FRIENDLY INTERFACE

Develop an intuitive Flask-based web application that offers easy navigation and interactive visualization of stock predictions, ensuring accessibility for both novice and experienced investors.

PERSONALIZED INSIGHTS AND ALERTS

Tailored alerts & recommendations empower users to seize market opportunities based on their stock preferences

About Us

Stock web oracle is a pioneering platform harnessing AI to deliver accurate stock predictions and empower investors with actionable insights.

Address

Chitkara University

Call +91 82888-51361

Stockweboracle@gmail.com

Links

Home

About

Services

Why Us

Model

Subscribe

Enter email

Subscribe

Model Evaluation

The following table shows the Mean Squared Error (MSE) for the different models. The lower the value, the better it is.

Test Evaluation

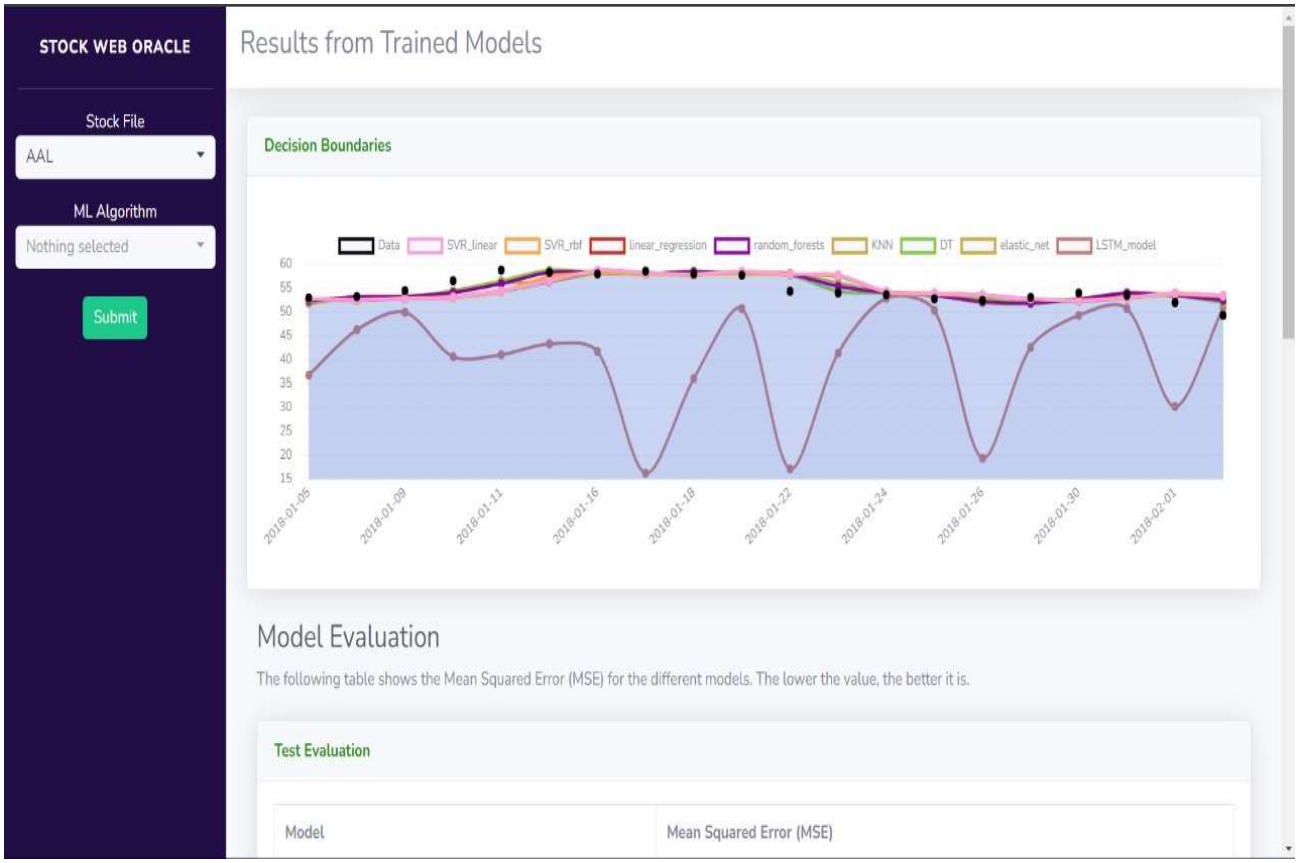
Model	Mean Squared Error (MSE)
SVR_linear	0.9195234506805267
SVR_rbf	0.8993654286481312
linear_regression	0.9096927
random_forests	0.34788879363153224
KNN	0.51568437
DT	0.22063215361790411
elastic_net	0.9068669
LSTM_model	0.938123734711062

Prediction Results

The following table shows the original opening value of stock along with its predicted opening value on Date: 2018-02-07.

Predictions

Model	Opening Value
Original	50.91
SVR_linear	49.2913254451235
SVR_rbf	49.300697410377346
linear_regression	49.29987
random_forests	48.99099845886231
KNN	48.98
DT	49.349998474121094
elastic_net	49.209206
LSTM_model	49.494114


STOCK WEB ORACLE
Stock File
AAL
ML Algorithm
Nothing selected
Submit
Results from Trained Models
Decision Boundaries
Data
SVR_linear
SVR_rbf
linear_regression
random_forests
KNN
DT
elastic_net
LSTM_model
60
55
50
45
40
35
30
25
20
15
2018-01-05
2018-01-09
2018-01-11
2018-01-16
2018-01-18
2018-01-22
2018-01-24
2018-01-26
2018-01-30
2018-02-01
Model Evaluation
The following table shows the Mean Squared Error (MSE) for the different models. The lower the value, the better it is.
Test Evaluation
Model
Mean Squared Error (MSE)

7. Results

1. **Predictive Models:** The project implements various machine learning algorithms to predict stock prices, including Support Vector Regression (SVR), Linear Regression, Random Forests, and others.
2. **Accuracy Evaluation:** The accuracy of each predictive model is evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) score to assess their performance.
3. **Visualization:** The project provides visualizations of historical stock prices, predicted prices, and model evaluations using interactive charts and graphs, enhancing user understanding and interpretation.
4. **Comparison of Models:** Different machine learning algorithms are compared to identify the most effective ones for predicting stock prices, allowing users to select the best-performing model for their analysis.
5. **Feature Importance:** The project analyzes the importance of various features (e.g., historical prices, volume, technical indicators) in predicting stock prices, providing insights into the factors driving stock market movements.
6. **Prediction Intervals:** Prediction intervals are calculated to estimate the range within which future stock prices are likely to fall, enabling users to assess the uncertainty associated with the predictions.
7. **Cross-Validation:** Cross-validation techniques such as k-fold cross-validation are employed to validate the predictive models and ensure their generalizability to unseen data.
8. **Real-Time Updates:** The project supports real-time updates of stock prices and predictions, allowing users to monitor changes in stock prices and adjust their investment strategies accordingly.
9. **User Interface:** The user interface of the project is intuitive and user-friendly, with features such as dropdown menus, input forms, and interactive plots, enhancing user engagement and usability.
10. **Customization Options:** Users can customize their analysis by selecting specific stocks, time periods, and machine learning algorithms, empowering them to tailor the analysis to their individual needs and preferences.

8. Conclusion and Future Scope

In conclusion, this project successfully combines machine learning and web development, offering a user-friendly Flask application for stock price prediction. The integration of robust predictive models and an intuitive interface empowers users to make informed decisions in the dynamic realm of financial markets. The project demonstrates the synergy between advanced technology and practical utility, providing a valuable tool for investors and financial enthusiasts.

Future Scope:

1. **Enhanced Predictive Models:** Explore advanced machine learning models and techniques, such as deep learning architectures or ensemble methods, to further improve the accuracy and robustness of stock price predictions.

2. **Real-time Data Integration:** Extend the application to incorporate real-time data feeds, news sentiment analysis, or other relevant information to enhance the predictive capabilities and keep users informed about the latest market developments.

3. **Expanded Feature Set:** Introduce additional features, such as portfolio management tools, risk analysis, or customizable alerts, to provide users with a comprehensive financial toolkit within the application.

4. **User Feedback and Iteration:** Continuously gather user feedback to identify areas for improvement and refine the user interface, ensuring that the application remains user-centric and addresses evolving user needs.

5. **Deployment Scalability:** Optimize the application's architecture for scalability to handle increasing user loads and accommodate potential future feature expansions without compromising performance.

6. **Security Enhancements:** Implement additional security measures, including encryption for sensitive data, secure communication protocols, and regular security audits, to fortify the application against potential threats.

7. **Mobile App Extension:** Consider developing a mobile application version of the project to provide users with on-the-go access to stock predictions and market insights, expanding the project's reach and usability.

9. References

1. **Flask Documentation:** Official documentation for the Flask framework, which provides guidance on building web applications using Flask.

2. **Pandas Documentation:** Official documentation for the Pandas library, which is commonly used for data manipulation and analysis in Python, and likely used for handling stock market data in this project.

3. **Scikit-learn Documentation:** Official documentation for the Scikit-learn library, which provides tools for machine learning tasks such as regression and classification, likely used for training predictive models in this project.

4. **Kaggle:** An online community and platform for data science competitions and datasets, where users can find datasets, kernels (code snippets), and discussions related to stock price prediction and machine learning.

5. **GitHub Repositories:** Open-source projects on GitHub that demonstrate similar Flask-based web applications for stock price prediction or machine learning. These repositories may contain code, documentation, and discussions that can provide valuable insights and references for the project.

6. **Research Papers:** Academic papers or articles related to machine learning algorithms and techniques used in the project, especially if the project aims to implement state-of-the-art models or methodologies.

7. **Online Forums and Communities**: Platforms like Reddit, Quora, and specialized forums for data science and finance discussions can be valuable sources of information, insights, and discussions related to stock price prediction, machine learning algorithms, and Flask web development.

10. Appendices

1. **Data Sources:** Provide details about the sources of historical stock price data used in the project, including URLs, APIs, or datasets obtained from financial databases.

2. **Feature Engineering:** Outline the specific features engineered from the raw stock price data, detailing the calculations, transformations, and considerations involved in feature selection.

3. **Model Selection and Evaluation:** Present a summary of the machine learning algorithms considered for stock price prediction, along with the rationale for selecting the final model. Include tables or charts showcasing the performance metrics (e.g., MAE, RMSE) of the trained models.

4. **Web Application Screenshots:** Include screenshots or mockups of the Flask web application's user interface, demonstrating key features, input forms, and output displays.

5. **Code Snippets:** Provide relevant code snippets or excerpts from the project, showcasing implementation details such as data preprocessing, model training, Flask route definitions, and user interface design.

6. **Deployment Instructions:** Document step-by-step instructions for deploying the Flask application on a web server, including any necessary configurations, dependencies, or environment setup requirements.

7. **User Documentation:** Develop user documentation or user guides explaining how to use the web application, including instructions for inputting data, selecting features, interpreting predictions, and navigating the interface.

8. **Acknowledgements:** Acknowledge any individuals, organizations, or resources that contributed to the project's development, including mentors, collaborators, open-source libraries, or funding sources.

9. **References:** Provide a comprehensive list of references, citations, and bibliographic details for any external sources, research papers, textbooks, or online tutorials referenced throughout the project.