Capstone Projects

COMPUTER SIMULATION USING MATLAB

Instructions:

1. **Get to know your project team:** I have formed random groups of 3 members each. Please liaise with your team as soon as possible and kickstart the project.
2. **Project Pick Time:** Choose the capstone project that excites you the most.
3. **Submission:** Submit your project deliverables via blackboard by Monday of Week 13. Ensure you include group member names & IDs.
4. **Presentation:** Be ready to present your project findings during Week 13 (slides encouraged)

Capstone Project 1: Optimizing Delivery Routes for a Food Delivery Service (using Optimization and Data Analysis)

Project Description:

Develop a system in MATLAB to optimize delivery routes for a food delivery service. The system should consider factors like distance, traffic patterns, and order delivery times to minimize delivery time and cost.

Key Components:

1. **Data Acquisition and Preprocessing:**
    - Collect data on delivery locations (addresses, coordinates), estimated travel distances, and historical traffic patterns (if available).
    - Clean and pre-process the data in MATLAB to ensure accuracy and consistency (e.g., handling missing values, converting addresses to coordinates).
2. **Distance Matrix Generation:**
    - Utilize MATLAB's built-in functions or external libraries to calculate the distance matrix between all delivery locations. This matrix will represent the travel time or cost between each pair of locations.
3. **Traffic Integration (Optional):**
    - If historical traffic data is available, explore methods to integrate it into the model. This could involve time-dependent distance matrices or penalty functions for congested routes.

4. **Optimization Algorithm Selection and Implementation:**
    - Choose an appropriate optimization technique to find the optimal delivery route. Here are some options:
        - **Nearest Neighbor:** A simple algorithm that visits the closest unvisited location at each step.
        - **Genetic Algorithms:** Mimic biological evolution to find an optimal solution through mutation and selection.
        - **Simulated Annealing:** Gradually explore the solution space to escape local optima and find a good solution.
    - Implement the chosen optimization algorithm in MATLAB to determine the route that minimizes total travel time or cost while considering all delivery locations.
5. **Visualization:**
    - Develop a visualization tool in MATLAB to represent the optimized delivery route on a map. This could involve plotting the route on a geographic map library or using a simpler visual representation of the delivery sequence.

Deliverables:

1. **MATLAB Script or Function:** A well-documented MATLAB script that takes delivery location data, distance matrix (and optionally traffic data) as input and outputs the optimized delivery route.
2. **Visualization:** A map or plot that visually represents the optimized delivery route for a given scenario.
3. **Report:** A report explaining the implemented optimization technique, its effectiveness in reducing delivery time and cost, and potential limitations of the model (e.g., not accounting for real-time traffic).

Grading Rubric:

Criteria	Excellent (25 pts)	Good (20 pts)	Satisfactory (15 pts)	Needs Improvement (10 pts)
Data Integration	Script effectively utilizes distance data, and optionally integrates traffic patterns (if available) to calculate realistic delivery times or costs.	Data integration is present, but some data might be missing or underutilized.	Script has limitations in data usage, or data may not be fully relevant to the optimization process.	Script lacks data integration or uses irrelevant data.
Optimization Approach	Effective selection and implementation of an optimization technique considering the problem and computational efficiency.	Optimization technique is chosen but may have limitations in effectiveness or implementation.	Optimization approach is present but lacks key elements or has significant limitations.	Optimization component is missing or not functional.
Visualization Clarity	Map or plot clearly communicates the optimized delivery route, including starting point, delivery locations, and travel path.	Visualization provides basic information about the route but may lack clarity or essential details.	Visualization is rudimentary or incomplete and doesn't effectively represent the route.	Visualization component is missing or not functional.

Report Quality	Report clearly explains the optimization technique, its impact on delivery efficiency (reduced time/cost), potential limitations of the model, and data considerations. The report is well-organized, concise, and uses proper grammar.	Report addresses key aspects but may lack depth in explaining the approach, results, or limitations.	Report has some content but lacks clarity, organization, or completeness.	Report is missing or does not address the required aspects.
Code Quality	Code is well-documented, modular, and easy to understand. Comments explain key functions and logic. Code adheres to good programming practices.	Code is documented, but comments might be limited or unclear. Code structure could be improved.	Code has some documentation, but overall quality and clarity need significant improvement.	Code is poorly documented or lacks documentation altogether. Code structure is difficult to follow.

Capstone Project 2: Modeling Forest Fire Spread (using Cellular Automata and Agent-Based Modeling)

Project Description:

Develop a simulation environment in MATLAB to model the spread of forest fires. You can explore two approaches:

- **Cellular Automata Model:** Simulate the forest as a grid of cells, where each cell represents a tree (healthy, burning, or burnt). Implement rules for fire propagation based on the state of neighboring cells.
- **Agent-Based Model:** Model individual trees as agents that can be healthy, burning, or burnt. Implement rules for fire spread based on the interaction between trees and environmental factors (wind).

Choose one approach to focus on for your project.

Key Components (Cellular Automata Approach):

1. **Forest Grid Creation:**
    - Define the size and resolution of the forest grid (number of rows and columns) in MATLAB.
    - Initialize the grid with cells representing healthy trees (dominant state) and a small number of burning trees (initial fire source).
2. **Fire Spread Rules:**
    - Implement rules for fire propagation based on the Moore neighborhood (neighboring cells in all eight directions).
    - A healthy tree can catch fire if a certain number of burning neighbors are present (e.g., two or more).
    - A burning tree transitions to burnt after a specific time step (representing the burning duration).
3. **Environmental Factors (Optional):**
    - Explore ways to integrate environmental factors like wind direction into the model.

- Wind could increase the probability of fire spread in the downwind direction by modifying the fire propagation rules.

4. **Simulation and Visualization:**
    - Develop a loop in MATLAB to iterate through time steps.
    - In each time step, update the grid based on the fire spread rules.
    - Utilize visualization tools in MATLAB to display the dynamic changes in the forest grid over time, highlighting healthy, burning, and burnt areas. This could involve animation or color-coded plots.

Key Components (Agent-Based Modeling Approach):

1. **Agent Definition:**
    - Define an agent class in MATLAB to represent individual trees within the forest.
    - Each agent can have attributes like health status (healthy, burning, burnt), location (coordinates within the forest), and flammability (a random value).
2. **Interaction and Fire Spread Rules:**
    - Implement rules for interaction between neighboring trees.
    - A healthy tree with high flammability has a higher chance of catching fire from a burning neighbor within a certain distance.
    - Burning trees transition to burnt after a specific time step.
3. **Environmental Factors:**
    - Consider wind as an environmental factor that affects fire spread.
    - Implement wind direction by modifying the interaction rules. For example, a burning tree is more likely to ignite healthy trees downwind.
4. **Simulation and Visualization:**
    - Develop a loop in MATLAB to iterate through time steps.
    - In each time step, agents update their state (healthy, burning, burnt) based on interaction rules and environmental factors.
    - Utilize visualization tools in MATLAB to display the dynamic changes in the forest over time, highlighting the location and health status of each tree (healthy, burning, burnt).

Deliverables:

1. **MATLAB Script or Function:** A well-documented MATLAB script that implements the chosen modeling approach (cellular automata or agent-based) to simulate forest fire spread.
2. **Visualization:** An animation or plot that visually represents the fire spread dynamics within the simulated forest environment.
3. **Report:** A report explaining the chosen modeling approach, factors considered in the simulation (e.g., wind direction - if applicable), and the observed fire spread patterns. The report should discuss the limitations of the model and potential applications (e.g., forest fire prediction, mitigation strategy evaluation).

Grading Rubric:

Criteria	Excellent (25 pts)	Good (20 pts)	Satisfactory (15 pts)	Needs Improvement (10 pts)
Modeling Approach	Clear and effective implementation of either cellular automata or agent-based modeling to simulate forest fire spread.	Modeling approach is chosen but may lack complexity or some key elements.	Modeling approach is present but has significant limitations or inaccuracies.	Modeling component is missing or not functional.
Factor Integration (if applicable)	The simulation considers relevant factors like fire propagation rules, environmental conditions (wind) in the chosen	Some factors are integrated into the simulation, but others might be missing or underutilized.	The simulation lacks essential factors or uses them inaccurately.	The simulation does not consider relevant factors for fire spread.

	modeling approach.	
Visualization Effectiveness	Animation or plot effectively communicates the dynamics of fire spread within the forest environment.	Visualization provides basic information about the fire spread but may lack clarity or

Capstone Project 3: Epidemic Modeling and Mitigation Strategies (using Agent-Based Modeling and Data Analysis)

Project Description:

Develop an agent-based simulation in MATLAB to model the spread of an infectious disease within a population. Analyze the effectiveness of different mitigation strategies (e.g., vaccination, social distancing) on controlling the outbreak.

Key Components:

1. **Agent Definition:**
    - Define an agent class in MATLAB to represent individuals within the population.
    - Each agent can have attributes like health status (susceptible, infected, recovered), location (coordinates within a simulated environment), and additional characteristics relevant to the disease (e.g., vaccination status).
2. **Disease Transmission Rules:**
    - Implement rules for disease transmission based on interactions between agents.
    - The probability of transmission can depend on factors like proximity, duration of contact, and individual attributes (e.g., vaccination status for some diseases).
3. **Disease Progression:**
    - Define a model for disease progression within infected agents. This could involve transitions between infected and recovered states after a specific duration or with a certain probability.
4. **Mitigation Strategies:**
    - Integrate functionalities to simulate different mitigation strategies:
        - **Vaccination:** Introduce a mechanism to vaccinate a certain percentage of the population at the beginning of the simulation or at specific intervals. Vaccinated agents may have reduced susceptibility or complete immunity.

        - **Social Distancing:** Modify interaction rules to reduce the frequency or proximity of contacts between agents, simulating social distancing measures.
5. **Simulation and Data Collection:**
    - Develop a loop in MATLAB to iterate through time steps, simulating the spread of the disease.
    - In each time step, agents update their health status based on disease transmission rules and progression.
    - Collect data on the number of susceptible, infected, and recovered individuals at each time step.
6. **Data Analysis and Visualization:**
    - Utilize MATLAB's data analysis tools to analyze the impact of different mitigation strategies on the course of the epidemic.
    - Generate plots or visualizations that represent:
        - The spread of the disease over time (number of infected individuals) under various scenarios (with and without mitigation strategies).
        - The effectiveness of mitigation strategies in reducing peak infection rates or total number of infected individuals.

Deliverables:

1. **MATLAB Script or Function:** A well-documented MATLAB script that implements the agent-based model for disease spread and allows for the simulation of different mitigation strategies.
2. **Data Analysis Report:** A report analyzing the simulation results. This should include visualizations of disease spread patterns under various scenarios (e.g., infection rate graphs) and a discussion on the effectiveness of the implemented mitigation strategies.
3. **Presentation (Optional):** A presentation summarizing the project, the model, and the key findings regarding mitigation strategies (can be delivered as slides or a recorded video).

Grading Rubric:

Criteria	Excellent (25 pts)	Good (20 pts)	Satisfactory (15 pts)	Needs Improvement (10 pts)
Model Functionality	Agent-based model effectively simulates disease transmission based on agent interactions and disease characteristics.	Model captures the essential elements of disease spread but may have limitations in complexity or accuracy.	Model functionality is present but lacks some key aspects or has significant limitations.	Model functionality is incomplete or not implemented.
Mitigation Strategies	Script allows for the simulation of various mitigation strategies (vaccination, social distancing) and their impact on the disease spread.	Mitigation strategies are included but may be limited in scope or functionality.	Mitigation strategies are present but lack key elements or have significant limitations.	Mitigation strategies are missing or not functional.
Data Analysis & Visualization	Data analysis effectively explores the impact of mitigation strategies on	Data analysis addresses key aspects but may lack depth or clarity in	Data analysis is limited or visualizations are ineffective in	Data analysis or visualizations are missing.

	disease spread. Clear visualizations effectively represent the simulation results.	presentation. Visualizations are present but might be rudimentary or unclear.	communicating results.	
Presentation (Optional)	Presentation (if provided) is clear, concise, and effectively communicates the project, model, and key findings.	Presentation is present but may lack clarity or organization.	Presentation is rudimentary or incomplete.	Presentation is missing.
Code Quality	Code is well-documented, modular, and easy to understand. Comments explain key functions and logic. Code adheres to good programming practices.	Code is documented, but comments might be limited or unclear. Code structure could be improved.	Code has some documentation, but overall quality and clarity need significant improvement.	Code is poorly documented or lacks documentation altogether. Code structure is difficult to follow.

Capstone Project 4: Smart Traffic Light Control (using System Dynamics and Optimization)

Project Description:

Develop a system dynamics model in MATLAB to simulate traffic flow at an intersection and design an optimization algorithm to improve traffic light timing for reduced congestion.

Key Components:

1. **System Model Definition:**
    - Identify the key elements and relationships involved in traffic flow at an intersection. These can include:
        - Arrival rates of vehicles from different directions.
        - Vehicle speeds within the intersection.
        - Queue lengths for each lane approaching the intersection.
        - Traffic light timings (green, yellow, red durations) for each direction.
2. **System Dynamics Equations:**
    - Develop a system of differential equations in MATLAB that represent the change in queue lengths and average waiting times based on arrival rates, service times (green light duration), and vehicle capacities.
3. **Traffic Light Control Strategy:**
    - Define a traffic light control strategy within the system dynamics model. This strategy can involve fixed timings, adaptive control based on real-time queue lengths (if data available), or a combination of both.
4. **Optimization Integration:**
    - Integrate an optimization technique (linear programming or genetic algorithms) to find the optimal traffic light timings that minimize average waiting time or total travel time for vehicles within the intersection. The optimization algorithm will manipulate the green light durations within the control strategy.
5. **Calibration and Validation (Optional):**

- If historical traffic data is available (e.g., arrival rates, queue lengths), use it to calibrate the model parameters (e.g., vehicle speeds) to ensure it reflects real-world traffic patterns.
- Validate the model's accuracy by comparing its simulation results with real-world traffic data (e.g., average waiting times).

6. **Simulation and Analysis:**
    - Develop a simulation loop in MATLAB to run the system dynamics model under different scenarios:
        - Baseline scenario with fixed traffic light timings.
        - Scenario with optimized traffic light timings obtained from the optimization algorithm.
    - Analyze the simulation results to compare average waiting times, queue lengths, and overall traffic flow efficiency between the baseline and optimized scenarios.

Deliverables:

1. **MATLAB Script or Function:** A well-documented MATLAB script that implements the system dynamics model, the traffic light control strategy, and the optimization algorithm for traffic light timing.
2. **Simulation Results:** Plots or visualizations that represent traffic flow patterns (queue lengths, waiting times) under different traffic light timing scenarios (before and after optimization).
3. **Report:** A report explaining the system dynamics model, the optimization approach, and the impact of optimized traffic light timings on traffic congestion. The report should discuss the limitations of the model and potential real-world implementation considerations (e.g., sensor data integration).

Grading Rubric:

Criteria	Excellent (25 pts)	Good (20 pts)	Satisfactory (15 pts)	Needs Improvement (10 pts)

Model Representation	System dynamics model effectively captures the key elements of traffic flow at an intersection, including arrival rates, queue lengths, and traffic light timings.	Model captures the essential elements but may lack some key aspects or have limitations in complexity.	Model functionality is present but lacks some key elements or has significant limitations.	Model functionality is incomplete or not implemented.
Optimization Integration	Optimization algorithm is effectively integrated with the system dynamics model to determine optimal traffic light timings.	Optimization component is present but may have limitations in effectiveness or implementation.	Optimization approach is present but lacks key elements or has significant limitations.	Optimization component is missing or not functional.
Calibration & Validation (if applicable)	Script demonstrates efforts to calibrate the model using real-world traffic data (if available) and	Calibration and validation attempts are present but may be limited in scope or effectiveness.	Calibration and validation are partially addressed or lacking justification.	Calibration and validation are missing.

	validates its accuracy.			
Results & Analysis	Simulation results clearly demonstrate the impact of optimized traffic light timings on traffic congestion (reduced waiting times). The report provides a clear analysis of the findings.	Results are presented but may lack clarity or depth in analysis of the impact of optimization.	Results are presented but lack clear interpretation or analysis.	Results are missing or unclear.
Code Quality	Code is well-documented, modular, and easy to understand. Comments explain key functions and logic. Code adheres to good programming practices.	Code is documented, but comments might be limited or unclear. Code structure could be improved.	Code has some documentation, but overall quality and clarity need significant improvement.	Code is poorly documented or lacks documentation altogether. Code structure is difficult to follow.

Capstone Project 5: Social Media Influence Analysis (using Agent-Based Modeling and Data Analysis)

Project Description:

Develop an agent-based model in MATLAB to simulate the spread of information and influence within a social network. Analyze how factors like user trust, content virality, and network structure impact the reach of information.

Key Components:

1. **Agent Definition:**
    - Define an agent class in MATLAB to represent users within a social network.
    - Each agent can have attributes like:
        - Information exposure (whether they have seen the information).
        - Trust levels towards other users (based on past interactions or reputation).
        - Sharing behavior (propensity to share information with others).
2. **Network Representation:**
    - Choose a method to represent the social network structure:
        - **Adjacency Matrix:** A matrix where rows and columns represent users, and entries indicate connections (e.g., 1 for a connection, 0 otherwise).
        - **Edge List:** A list of pairs of users connected in the network.
    - Consider using real-world social network data (if available) to represent the network structure for increased realism.
3. **Information Propagation Rules:**
    - Implement rules for how information spreads based on agent interactions and trust:
        - An agent can be exposed to information through connections in the network who have already shared it.

        - The probability of sharing the information further depends on the agent's trust in the source and the perceived content virality (e.g., interesting or newsworthy content).
4. **Simulation and Data Collection:**
    - Develop a simulation loop in MATLAB to iterate through time steps.
    - In each time step, agents update their information exposure status based on the propagation rules and their interactions with other agents in the network.
    - Collect data on the number of exposed users, the cascade of information flow (who shared with whom), and the saturation point (when information reaches a maximum number of users).
5. **Data Analysis and Visualization:**
    - Utilize MATLAB's data analysis tools to analyze the impact of different factors on information spread:
        - **User Trust:** Compare scenarios with varying levels of trust in the network (high vs. low) and observe its influence on information reach.
        - **Content Virality:** Simulate different content types with varying virality levels and analyze how it affects the spread and saturation point.
        - **Network Structure:** Explore different network structures (e.g., random graphs vs. small-world networks with high clustering) and compare their impact on information diffusion patterns (e.g., faster spread in small-world networks).
    - Generate plots or visualizations that represent:
        - The spread of information over time (number of exposed users) under various scenarios.
        - The cascade structure of information flow within the network.

Deliverables:

1. **MATLAB Script or Function:** A well-documented MATLAB script that implements the agent-based model for information diffusion within a social network. The script should allow for the simulation of different scenarios with varying trust levels, content types, and network structures.
2. **Data Analysis Report:** A report analyzing the simulation results. This should include visualizations of information diffusion patterns under various scenarios and a discussion on the factors that influence the reach and virality of information within the social network.
3. **Presentation (Optional):** A presentation summarizing the project, the model, and the key findings regarding the impact of trust, virality, and network structure on information diffusion (can be delivered as slides or a recorded video).

Grading Rubric:

Criteria	Excellent (25 pts)	Good (20 pts)	Satisfactory (15 pts)	Needs Improvement (10 pts)
Model Functionality	Agent-based model effectively simulates information diffusion based on user interactions, trust dynamics, and content characteristics.	Model captures the essential elements of information spread but may lack complexity in trust dynamics, content types, or network interactions.	Model functionality is present but lacks some key aspects of information diffusion (trust, content, or network) or has limitations in user behavior representation.	Model functionality is incomplete or not implemented, hindering simulation of information diffusion.
Factor Analysis	Script allows for the simulation of	Script allows for variation in at least	Script has limited functionality for	Script lacks functionality to analyze the

	various scenarios with different trust levels (high vs. low), content types (high vs. low virality), and network structures (e.g., random vs. small-world). Users can easily modify these parameters within the script.	two factors (trust, content, or network) but may have limitations in the number of scenarios or ease of modification.	exploring different factors affecting information diffusion. Users may struggle to modify parameters for various scenarios.	impact of different factors on information diffusion, or factors cannot be easily modified for scenario exploration.
Data Analysis & Visualization	Data analysis effectively explores the impact of trust, virality, and network structure on information diffusion. Clear and informative visualizations effectively represent the simulation results (spread over time, cascade structure).	Data analysis addresses key factors but may lack depth or clarity in presentation. Visualizations are present but might be rudimentary or unclear in conveying findings.	Data analysis is limited or visualizations are ineffective in communicating results. The impact of different factors is not clearly explored or visualized.	Data analysis or visualizations are missing. There is no exploration of how factors influence information diffusion.

Presentation (Optional)	Presentation (if provided) is clear, concise, and effectively communicates the project, model, and key findings regarding the impact of trust, virality, and network structure on information diffusion. Visual aids are well-integrated and enhance understanding.	Presentation is present but may lack clarity or organization in explaining the project, model, or findings. Visual aids may be unclear or not well-utilized.	Presentation is rudimentary or incomplete, lacking key information about the project, model, or findings. Visual aids are missing or ineffective.	Presentation is missing.
Code Quality	Code is well-documented, modular, and easy to understand. Comments explain key functions, logic behind trust dynamics, and user behavior representation. Code adheres to good programming practices.	Code is documented, but comments might be limited or unclear. Code structure could be improved.	Code has some documentation, but overall quality and clarity need significant improvement, making it difficult to understand the implementation.	Code is poorly documented or lacks documentation altogether. Code structure is difficult to follow.

Capstone Project 6: Stock Market Price Prediction with Machine Learning (using Time Series Analysis and Neural Networks)

Project Description:

Develop a machine learning model in MATLAB to predict stock market prices based on historical data. Explore different techniques like time series analysis and neural networks to evaluate their effectiveness in price prediction.

Key Components:

1. **Data Acquisition:**
    - Collect historical stock market data for a chosen stock or index. This data can include:
        - Closing prices
        - Trading volumes
        - Technical indicators (e.g., moving averages, relative strength index)
2. **Data Preprocessing:**
    - Clean and pre-process the collected data in MATLAB:
        - Handle missing values (e.g., imputation techniques).
        - Normalize or scale the data (if necessary) to ensure features are on a similar scale.
        - Convert the data into a format suitable for machine learning models (e.g., time series for past closing prices as input, future closing price as target variable).
3. **Time Series Analysis:**
    - Implement a time series forecasting model in MATLAB, such as:
        - **Autoregressive Integrated Moving Average (ARIMA):** A statistical model that uses past values of the time series to predict future values.
        - **Exponential Smoothing:** A technique that assigns weights to past observations, with more recent data having higher weights.
    - Train the time series model on historical data and evaluate its performance on a hold-out test set. Metrics like Mean Squared Error

(MSE) or Mean Absolute Error (MAE) can be used to assess prediction accuracy.

4. **Neural Network Exploration:**
    - Implement a neural network model in MATLAB for stock price prediction. Popular choices include:
        - **Long Short-Term Memory (LSTM) Networks:** A type of recurrent neural network that can effectively learn long-term dependencies in time series data.
        - **Convolutional Neural Networks (CNNs):** If technical indicators are included as features, CNNs can be used to extract relevant patterns from this data.
    - Train the neural network model on the pre-processed data and evaluate its performance on the test set. Compare the prediction accuracy with the time series model using the same metrics.
5. **Model Comparison and Analysis:**
    - Analyze the performance of both the time series and neural network models.
    - Discuss the advantages and disadvantages of each approach in the context of stock price prediction.
    - Consider factors like model complexity, interpretability of results, and potential limitations of each model.

Deliverables:

1. **MATLAB Script or Function:** A well-documented MATLAB script that implements both the time series and neural network models for stock price prediction. The script should allow for data pre-processing, model training, and evaluation.
2. **Performance Analysis Report:** A report comparing the performance of the time series and neural network models. This should include:
    - Evaluation metrics (MSE, MAE) for both models.

- Discussion on the effectiveness of each approach in predicting stock prices.
- Limitations of the models and potential areas for improvement (e.g., incorporating additional features, hyperparameter tuning).

3. **Presentation (Optional):** A presentation summarizing the project, the implemented models, and the comparative analysis of their performance in stock price prediction (can be delivered as slides or a recorded video).

Grading Rubric:

Criteria	Excellent (25 pts)	Good (20 pts)	Satisfactory (15 pts)	Needs Improvement (10 pts)
Data Acquisition & Preprocessing	Script effectively retrieves and preprocesses relevant stock market data for machine learning models.	Script retrieves data but may lack some essential preprocessing steps.	Data acquisition or preprocessing is limited or incomplete.	Data is missing or not suitable for machine learning analysis.
Model Implementation	Script implements both a time series model (ARIMA or similar) and a neural network model (LSTM or CNN) for stock price prediction.	Script implements one model (time series or neural network) but may lack the other.	Script has limited functionality for model implementation.	Script lacks implementation of machine learning models.

Model Evaluation & Comparison	Script effectively evaluates the performance of both models using appropriate metrics (MSE, MAE) and provides a clear comparison of their prediction accuracy.	Script evaluates model performance but may lack a thorough comparison or use inappropriate metrics.	Model evaluation or comparison is limited or unclear.	Model evaluation or comparison is missing.
Analysis & Interpretation	Report provides a clear analysis of the results, discussing the strengths and weaknesses of each model in the context of stock price prediction.	Report addresses model performance but lacks depth in analysis or interpretation.	Report has limited analysis or discussion on the model performance.	Report lacks analysis or interpretation of the results.