University of Duisburg-Essen
Chair of Dynamics and Control
Univ.-Prof. Dr.-Ing. Dirk Söffker

Project work

Classification - Traffic Characteristic 1
Regression - Traffic Characteristic 2

Dhruv Patel
3181960

Advisor

Jonathan Liebeton, M.Sc.
Univ.-Prof. Dr.-Ing. Dirk Söffker

September 2024

Versicherung an Eides Statt

Ich versichere an Eides statt durch meine untenstehende Unterschrift,

- dass ich die vorliegende Arbeit - mit Ausnahme der Anleitung durch die Betreuer - selbstständig ohne fremde Hilfe angefertigt habe und
- dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus fremden Quellen entnommen sind, entsprechend als Zitate gekennzeichnet habe und
- dass ich ausschließlich die angegebenen Quellen (Literatur, Internetseiten, sonstige Hilfsmittel) verwendet habe und
- dass ich alle entsprechenden Angaben nach bestem Wissen und Gewissen vorgenommen habe, dass sie der Wahrheit entsprechen und dass ich nichts verschwiegen habe.

Mir ist bekannt, dass eine falsche Versicherung an Eides Statt nach §156 und nach §163 Abs. 1 des Strafgesetzbuches mit Freiheitsstrafe oder Geldstrafe bestraft wird.

Ort, Datum

Unterschrift

Urheberrechtsvereinbarung

Die vorliegende Arbeit entstand unter intensiver Mitwirkung der genannten, betreuenden Personen im Rahmen von Forschungsarbeiten im Lehrstuhl Steuerung, Regelung und Systemdynamik (SRS) der Universität Duisburg-Essen. Die in dieser Arbeit enthaltenen Lösungen und Ideen unterliegen daher nicht nur dem Urheberrecht der zu qualifizierenden Person, sondern dem genannten Personenkreis gemeinsam. Die persönliche, private sowie die Nutzung im Forschungskontext des Lehrstuhls SRS ist hiervon unbenommen.

Ort, Datum

Unterschrift

Contents

List of Figures

List of Tables

1 Introduction

This paper explores the application of machine learning (ML) techniques to traffic characteristics, focusing on how these methods can enhance the analysis and prediction of traffic conditions. ML offers powerful tools for both classification and regression tasks in traffic management. Classification involves predicting traffic conditions based on real-world data, such as traffic speed measurements, and categorizing them into classes like Heavy, Light, Free Flow, and Stopped/Signal/Starting. The goal is to implement and evaluate ML models for accurate classification and assess their performance using various metrics. Conversely, regression aims to forecast future traffic speed by analyzing historical data, employing methods such as neural networks and linear regression. This task involves predicting future speeds based on past measurements, with performance evaluated through training and testing losses. By leveraging these ML techniques, this paper addresses practical scenarios in traffic management and control, providing valuable insights for improving traffic flow and safety.[AY20]

1.1 Traffic Characteristic 1 - Classification

The classification of "traffic data" involves applying machine learning techniques to predict traffic conditions based on real-world data. The dataset consists of traffic speed measurements (in Km/Hr) collected over 10 days, with labels assigned to 10-second segments, representing four classes: Heavy, Light, Free Flow, and Stopped/Signal/Starting.

The goal is to implement machine learning models to accurately classify these traffic conditions and evaluate their performance using various scoring metrics. This task enables a direct application of machine learning methods to practical scenarios in traffic management and control.

1.2 Traffic Characteristic 2 - Regression

This regression task aims to predict future traffic speed using two approaches: a neural network and linear regression. Using speed data from "26.8.2023 afternoon" as training data, the goal is to forecast 1-second future speed based on past 5-second intervals. The data will be split 80% - 20% for training and testing, with the "27.8.2023 afternoon" data serving as the test set.

Performance will be evaluated by calculating training and testing losses. Additionally, the second dataset includes raw time-series data that requires pre-processing and feature extraction before applying machine learning methods.

2 Traffic Characteristic 1 - Classification

For the classification of traffic conditions, various machine learning algorithms were applied to the dataset, which includes traffic speed measurements (in Km/Hr) over 10 days, with 10-second segments labeled into four categories: Heavy, Light, Free Flow, and Stopped/Signal/Starting.

2.1 Algorithms to be compared

- **Decision Tree**: This model constructs a tree-like structure where each node represents a decision based on feature values. It provides a clear, interpretable classification by following the branches of the tree to make predictions.
- **K-Nearest Neighbors (KNN)**:KNN classifies data based on the majority label of the 'k' nearest neighbors, using proximity for predictions.
- **Naive Bayes**: Uses probabilistic calculations based on Bayes' theorem with the assumption of feature independence.
- **Support Vector Machine (SVM)**: SVM finds the optimal hyperplane that maximizes the margin between different classes in the feature space. It is effective in high-dimensional spaces and aims to create a clear boundary between classes. [Woo23]

2.2 Scoring Matrices

- **Accuracy**: Measures the proportion of correctly classified instances out of all instances. It provides a general indication of the model's overall performance.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

  - TP = True Positives
  - TN = True Negatives
  - FP = False Positives

  - FN = False Negatives

- **Precision**: Indicates the proportion of true positives among all instances classified as positive. It reflects the model's ability to avoid false positives, which is crucial when the cost of false positives is high.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall**: Shows the proportion of true positives among all actual positive instances. It assesses the model's ability to identify all relevant instances, emphasizing the importance of capturing as many true positives as possible.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score**: The harmonic mean of precision and recall, offering a balanced evaluation metric. It is particularly useful when there is a need to balance precision and recall. [Kal23]

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.3 Matlab Code

This section describes the MATLAB code used for implementing and evaluating machine learning classifiers on traffic data. The code is detailed below, with each step explained in separate subsections.

The MATLAB code performs the following tasks:

1. **Load and Initialize Data:**
   - `sheets` retrieves the names of all sheets in the `"Traffic_Data.xlsx"` file.
   - `opt` defines the import options for reading data from the Excel file.
   - `featureData` is initialized as an empty array to store merged data.
2. **Data Merging:**
   - The code loops through each sheet (except the last one) and appends the data to `featureData`.
3. **Prepare Features and Labels:**
   - `X` extracts features from the merged `featureData`.
   - `Y` converts the label column to a categorical variable for classification.
4. **Train-Test Split:**
   - `cv` performs an 80%-20% holdout split of the data.
   - `X_train` and `y_train` store the training features and labels.
   - `X_test` and `y_test` store the testing features and labels.
5. **Classifier Setup:**
   - `classifiersNames` lists the classifier types: Decision Tree, k-Nearest Neighbors, Naive Bayes, and Support Vector Machine.
   - `classifiers` contains function handles to initialize and train each classifier.
   - The setup allows easy addition of new models by inserting new function handles into the `classifiers` cell array.
6. **Model Training and Evaluation:**
   - The code loops through each classifier, trains it using `X_train` and `y_train`, and makes predictions on `X_test`.

- Metrics including accuracy, precision, recall, and F1 score are calculated and stored in `accuracyResults`, `precisionResults`, `recallResults`, and `f1ScoreResults`.
- Confusion matrices for each classifier are stored in `confusionMat`.

7. **Display Results:**
   - Displays tables showing accuracy, precision, recall, and F1 score for each classifier.
   - Confusion matrices for each classifier are displayed separately.
8. **Feature Analysis (Optional):**
   - Due to the clear decision boundaries observed in the dataset, where feature values for different classes do not overlap, the classifiers achieved 100% accuracy.
   - This perfect classification accuracy made it challenging to discern meaningful differences in model performance.
   - Therefore, the Feature Analysis part, was included to investigate the feature distributions and confirm the dataset characteristics contributing to the observed performance.

This MATLAB code effectively preprocesses data, trains various classifiers, and evaluates their performance using key metrics. It also provides an optional analysis of features, enhancing the understanding of data characteristics.

2.4 Results

The performance of four different classifiers—Decision Tree, k-Nearest Neighbors, Naive Bayes, and Support Vector Machine—was evaluated using the following scoring matrices: Accuracy, Precision, Recall, and F1 Score. The results are summarized in the tables below.

2.4.1 Confusion Matrix

$$\text{Decision Tree} = \begin{bmatrix} 8460 & 0 & 0 & 0 \\ 0 & 25077 & 0 & 0 \\ 0 & 0 & 13815 & 0 \\ 0 & 0 & 0 & 22291 \end{bmatrix}$$

$$\text{k-Nearest Neighbors} = \begin{bmatrix} 8460 & 0 & 0 & 0 \\ 0 & 25077 & 0 & 0 \\ 0 & 0 & 13815 & 0 \\ 0 & 0 & 0 & 22291 \end{bmatrix}$$

$$\text{Naive Bayes} = \begin{bmatrix} 8460 & 0 & 0 & 0 \\ 0 & 25077 & 0 & 0 \\ 0 & 0 & 13815 & 0 \\ 0 & 0 & 0 & 22291 \end{bmatrix}$$

$$\text{Support Vector Machine} = \begin{bmatrix} 8460 & 0 & 0 & 0 \\ 0 & 25077 & 0 & 0 \\ 0 & 0 & 13815 & 0 \\ 0 & 0 & 0 & 22291 \end{bmatrix}$$

2.4.2 Accuracy

Table 2.1: Accuracy

ClassifierID	Accuracy
Decision Tree	100.00
k-Nearest Neighbors	100.00
Naive Bayes	100.00
Support Vector Machine	100.00

2.4.3 Precision, Recall, F1 Score

Table 2.2: Precision

ClassifierID	Heavy	Light	Free flow	Stopped/Signal/Starting
Decision Tree	1.00	1.00	1.00	1.00
k-Nearest Neighbors	1.00	1.00	1.00	1.00
Naive Bayes	1.00	1.00	1.00	1.00
Support Vector Machine	1.00	1.00	1.00	1.00

Table 2.3: Recall

ClassifierID	Heavy	Light	Free flow	Stopped/Signal/Starting
Decision Tree	1.00	1.00	1.00	1.00
k-Nearest Neighbors	1.00	1.00	1.00	1.00
Naive Bayes	1.00	1.00	1.00	1.00
Support Vector Machine	1.00	1.00	1.00	1.00

Table 2.4: F1 Score

ClassifierID	Heavy	Light	Free flow	Stopped/Signal/Starting
Decision Tree	1.00	1.00	1.00	1.00
k-Nearest Neighbors	1.00	1.00	1.00	1.00
Naive Bayes	1.00	1.00	1.00	1.00
Support Vector Machine	1.00	1.00	1.00	1.00

2.4.4 Feature Analysis

Table 2.5: Feature Analysis Results

Mean	Variance	Median	MaxVal	MinVal	Label
1.4193	3.2321	0	0	5	Stopped/Signal/Starting
12.343	18.293	12	6	20	Heavy
27.26	18.41	27	21	35	Light
44.155	42.572	43	36	75	Free flow

2.5 Conclusion

All classifiers achieved identical performance across all metrics due to the clear decision boundaries present in the dataset. The data was well-separated into distinct classes, which allowed all classifiers to perfectly differentiate between different traffic conditions, resulting in 100% accuracy, precision, recall, and F1 scores for each label. The confusion matrices further confirm that each classifier correctly identified all instances without any misclassification.

3 Traffic Characteristic 2 - Regression

Traffic management and analysis are critical components in urban planning and transportation systems. With increasing vehicle numbers and complex traffic patterns, understanding and predicting traffic characteristics are essential for optimizing road usage, reducing congestion, and improving safety. Traffic characteristics include metrics such as traffic flow, vehicle speed, and congestion levels, which can be analyzed to develop effective management strategies.[AY20]

Traffic regression analysis addresses several key issues related to traffic management:

- **Vehicle Speed**: Vehicle speed is a significant factor in determining traffic efficiency and safety. Variations in speed can indicate congestion, road conditions, or driver behavior. Regression analysis can help predict speed variations and identify factors contributing to speed changes.
- **Congestion Analysis**: Traffic congestion is a major concern in urban areas, leading to delays and increased travel times. By analyzing congestion data through regression models, it is possible to identify trends, forecast congestion levels, and evaluate the effectiveness of traffic management interventions.
- **Impact of Traffic Signals and Road Infrastructure**: Traffic signals, road construction, and other infrastructure changes can affect traffic patterns. Regression analysis can be used to assess the impact of these changes on traffic flow, speed, and congestion, helping planners make informed decisions.
- **Accident and Incident Analysis**: Traffic accidents and incidents can significantly disrupt traffic flow and contribute to congestion. Regression models can analyze historical accident data to predict the likelihood of future incidents and their impact on traffic characteristics.

Given the complex interplay of factors influencing traffic dynamics, accurate prediction and analysis are vital. Techniques such as Linear Regression, polynomial regression, and time-series analysis are employed to model and forecast traffic characteristics based on historical and real-time data. These models provide valuable insights for improving traffic efficiency and safety.

In this study, we focus on traffic regression using both Linear Regression and Neural Network models. Linear Regression offers a fundamental approach by establishing relationships between traffic metrics and influencing variables such as time of day and weather conditions. Neural Networks provide a more sophisticated method, capable of modeling

complex, non-linear relationships in traffic data. These models can enhance prediction accuracy and offer deeper insights into traffic behavior.

The challenge in traffic regression lies in the variability of traffic patterns, influenced by factors such as peak hours, special events, and weather conditions. Additionally, the quality of available data significantly impacts model performance. This study aims to compare the effectiveness of Linear Regression and Neural Networks in predicting traffic characteristics, providing insights for traffic management and urban planning.

By employing these methodologies, this analysis will evaluate their strengths and limitations, offering a comprehensive understanding of traffic dynamics and supporting the development of more efficient traffic management solutions.[Deekshetha22]

3.1 Data Preprocessing

Effective data preprocessing is crucial for preparing time-series data for machine learning, improving model performance, and ensuring accurate predictions. This section outlines the preprocessing steps applied to traffic speed data, including various filtering techniques and additional considerations.

3.1.1 Data Extraction

Data was extracted from CSV files containing traffic speed measurements, split into training and test datasets, and organized for preprocessing.

3.1.2 Filter Design and Application

Several filtering techniques were explored to enhance data quality:

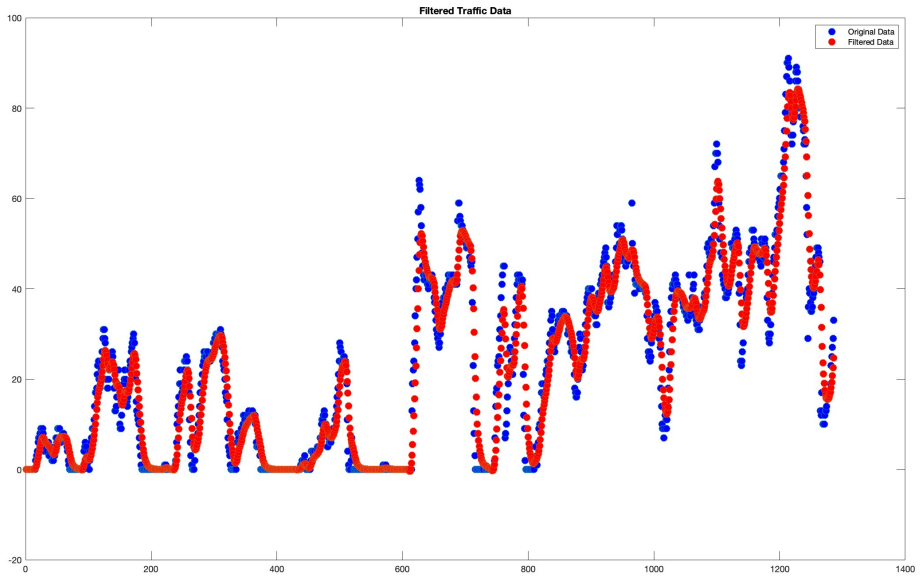
- **Wavelet Denoising:** This technique removes noise from the data across different scales, isolating and removing noise while preserving the underlying signal.[Gol19]
- **Savitzky-Golay Filter:** Applied to smooth the data while preserving important features. This filter uses polynomial fitting to smooth data points and retain critical features and trends.[Per03]
- **Exponential Moving Average (EMA):** Prioritizes recent data points by emphasizing recent observations, making it useful for real-time monitoring and short-term forecasting.

- **Low-Pass Butterworth Filter:** Designed to smooth the data by removing high-frequency noise. This filter reduces high-frequency components while balancing noise reduction with signal preservation. [Shi11]

3.1.3 Preprocessing Combinations

Various filter combinations were tested:

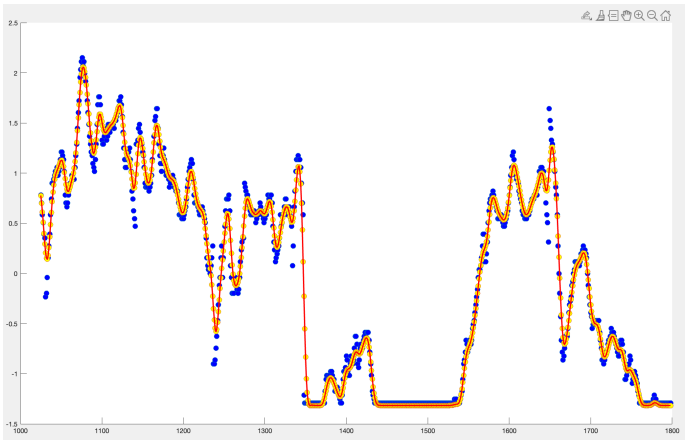
- **Wavelet Denoising + Savitzky-Golay + Exponential Moving Average (EMA):** This combination starts with wavelet denoising to remove noise, followed by Savitzky-Golay for feature preservation and smoothing, and EMA to emphasize recent trends.



(a) Wavelet Denoising + Savitzky-Golay + EMA

Figure 3.1: Comparison of data preprocessing methods using Wavelet Denoising, Savitzky-Golay filtering, and Exponential Moving Average (EMA). The figure illustrates the effectiveness of each method in removing noise, preserving features, and emphasizing recent trends.

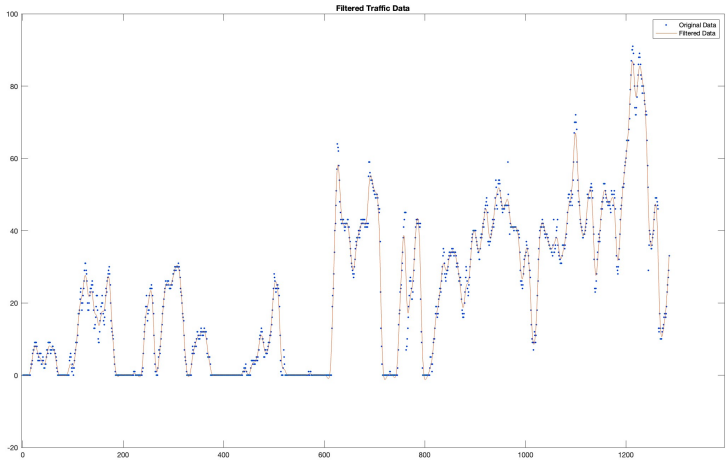
- **Exponential Moving Average (EMA) + Gaussian Smoothing:** Combines EMA for recent trends with Gaussian Smoothing for additional noise reduction.



(a) EMA + GS

Figure 3.2: Effectiveness of combining Exponential Moving Average (EMA) with Gaussian Smoothing.

- **Low-Pass Filter:** Selected for its effective noise reduction and smooth representation of data.



(a) Low Pass Filter

Figure 3.3: Performance of the Low-Pass Filter in noise reduction and data smoothing.

3.1.4 Normalization

Post-filtering, normalization was applied to standardize the data, ensuring consistency across features.

3.1.5 Feature and Label Generation

The sliding window approach is used to capture the temporal patterns in traffic speed data, which is inherently sequential. By considering a fixed window of past speed observations (e.g., 5 seconds) to predict the next time point (1 second ahead), we effectively capture how past values influence future values. This method generates multiple overlapping segments, increasing the training data size, which helps the model learn robustly from limited data.

As shown in Figure 3.5, the sliding window approach captures short-term changes in traffic speed, making it ideal for real-time predictions. Its flexibility allows for adjusting the window size to optimize model performance. This method transforms raw speed data into meaningful features, enhancing the accuracy of both neural network and linear regression models. [Salih24]

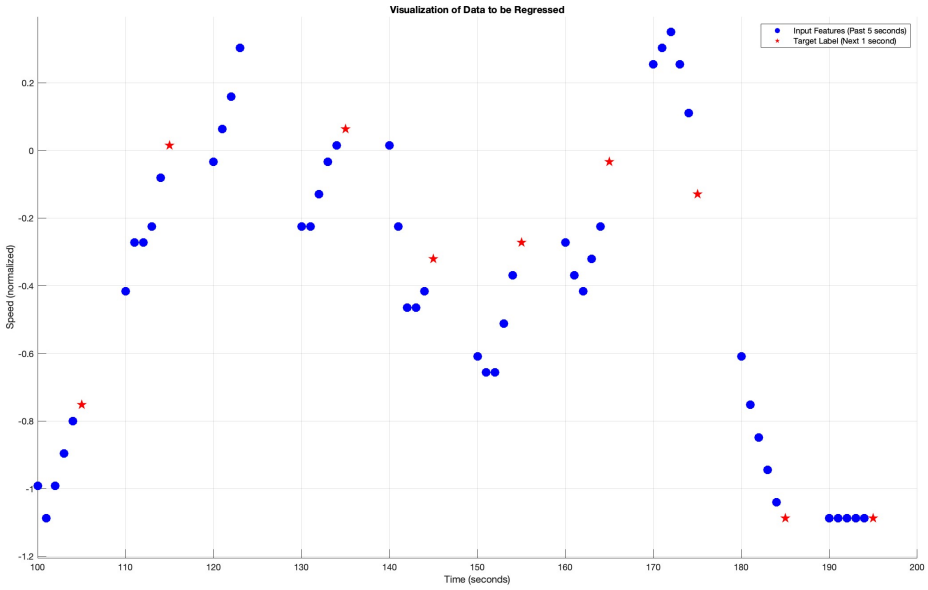
3.1.6 Train-Validation Split

The dataset was divided into training and validation sets to assess model performance and generalizability:

- **Purpose:** Ensure the model is trained on a subset of data and validated on another to evaluate its performance and avoid overfitting.
- **Explanation:** The training set was used to fit the model, learn patterns, and adjust parameters. The validation set was used to evaluate the model's performance and ensure it generalizes well to new, unseen data.

  The goal of this split is to gauge the model's performance and make necessary adjustments before testing it on a separate test set. This step is part of the overall data preparation process, setting the stage for effective model training and evaluation.

Data preprocessing involved testing various noise reduction methods, including combinations of Wavelet Denoising, Savitzky-Golay, EMA, and Gaussian Smoothing. After evaluation, a low-pass Butterworth filter was chosen for its effectiveness in removing noise



(a) Sliding Window approach for Feature Extraction

Figure 3.4: Illustration of the sliding window approach used for feature and label generation. The figure depicts how past speed observations within a fixed window (e.g., 5 seconds) are used to predict the next time point (1 second ahead). This technique effectively captures temporal patterns and transforms raw data into meaningful features for improved model accuracy.

while preserving key data features. The data was then normalized, and a sliding window approach was employed to generate features and labels that capture temporal patterns. This comprehensive preprocessing strategy aimed to enhance the performance and accuracy of both Neural Network and Linear Regression models.

3.2 MATLAB Code Overview

The MATLAB code is organized into several key sections. Each section is detailed below.

1. **Load and Initialize Data:**

   (a) Load training data from "Speed data-20240901/26.08.2023.csv" and testing data from "Speed data-20240901/27.08.2023.csv".

2. **Preprocessing:**
   (a) **Filtering:**
      i. Apply a Butterworth low-pass filter to remove high-frequency noise from the speed data.
      ii. Use a cutoff frequency of 0.08 Hz.
   (b) **Clipping:**
      i. Clip negative values in the speed data to zero.
   (c) **Normalization:**
      i. Normalize the filtered speed data to ensure a consistent range of values.
3. **Feature and Label Generation:**
   (a) Generate features and labels using a 5-second sliding window approach.
      i. **Features:** Past 5 seconds of speed data.
      ii. **Labels:** Speed data for the subsequent second.
   (b) Create training and test sets for the models.
4. **Train-Test Split:**
   (a) Perform an 80%-20% split to separate training and validation datasets.
5. **Model Training and Evaluation:**
   (a) **Linear Regression:**
      i. Train a Linear Regression model using ridge regularization.
      ii. Calculate training and validation losses.
      iii. Evaluate the model on the test set and calculate Mean Absolute Error (MAE).
      iv. Plot results comparing original test data, filtered data, and model predictions.
   (b) **Neural Network:**
      i. Define the network architecture with LSTM layers.
      ii. Set training options including learning rate, regularization, and epochs.
      iii. Train the Neural Network and evaluate its performance on the test set.
      iv. Plot results comparing original test data, filtered data, and model predictions.

6. **Cross-Validation:**
   (a) **Linear Regression:**
      i. Perform K-fold cross-validation to assess model generalizability and compute average MAE.
   (b) **Neural Network:**
      i. Use K-fold cross-validation to ensure robust performance evaluation and calculate average MAE.
7. **Display Results:**
   (a) Present tables showing training and validation losses for both Linear Regression and Neural Network models.
   (b) Display plots of predictions versus actual values for visual comparison.

3.3 Neural Network Architecture and Training Options

3.3.1 Network Architecture

The Neural Network is constructed with the following layers:

1. **Input Layer:**
   - `sequenceInputLayer(inputSize)` defines the input layer, where `inputSize` represents the number of features in the input sequences.
2. **LSTM Layer:**
   - `lstmLayer(numHiddenUnits, 'OutputMode', 'sequence')` adds an LSTM layer with `numHiddenUnits` units.
   - The `'OutputMode', 'sequence'` parameter ensures that the output is a sequence of values.
3. **Dropout Layer:**
   - `dropoutLayer(0.3)` applies dropout with a probability of 0.3 to prevent overfitting.
4. **Fully Connected Layer:**

- `fullyConnectedLayer(numClasses)` is a dense layer with `numClasses` neurons that outputs the network's final predictions.

5. **Regression Layer:**
    - `regressionLayer` is used for regression tasks to compute the loss between predicted and actual values.

3.3.2 Training Options

The training options are specified as follows:

1. **Optimizer:**
    - `'adam'` specifies the Adam optimizer, an adaptive learning rate optimization algorithm.
2. **Initial Learning Rate:**
    - `'InitialLearnRate', 0.02` sets the initial learning rate to 0.02.
3. **L2 Regularization:**
    - `'L2Regularization', 0.0001` applies L2 regularization with a coefficient of 0.0001 to prevent overfitting.
4. **Epochs:**
    - `'MaxEpochs', 100` specifies that training will run for a maximum of 100 epochs.
5. **Mini-Batch Size:**
    - `'MiniBatchSize', 20` sets the mini-batch size to 20.
6. **Shuffle:**
    - `'Shuffle', 'every-epoch'` ensures the training data is shuffled at the beginning of each epoch.
7. **Plots:**
    - `'Plots', 'none'` disables plotting the training progress.

8. **Validation Frequency:**
    - `'ValidationFrequency', 20` sets the validation frequency to every 20 iterations.
9. **Verbose:**
    - `'Verbose', false` disables detailed output during training.
10. **Learning Rate Schedule:**
    - `'LearnRateSchedule', 'piecewise'` indicates a piecewise learning rate schedule.
    - `'LearnRateDropFactor', 0.9` reduces the learning rate by a factor of 0.9.
    - `'LearnRateDropPeriod', 10` specifies that the learning rate will drop every 10 epochs.
11. **Validation Patience:**
    - `'ValidationPatience', 20` sets the patience for early stopping to 20 iterations without improvement in validation loss.

3.4 Regression Results

3.4.1 Performance Metrics

In this section, the performance of two regression models, namely Linear Regression and Neural Network, is evaluated. The evaluation is based on several key performance metrics such as Training Loss, Validation Loss, Test Loss (measured by the Mean Absolute Error or MAE), and Cross-Validation MAE. These metrics provide a comprehensive understanding of how well each model generalizes from the training data to unseen test data.

- **Training Loss:** This represents how well the model fits the training data. A lower training loss indicates that the model has learned the underlying relationships in the training data.
- **Validation Loss:** This shows the performance of the model on the validation set, providing insights into how well the model generalizes to unseen data.
- **Test Loss (MAE):** The Mean Absolute Error (MAE) on the test set reflects the model's accuracy when predicting new, unseen data.
- **Cross-Validation MAE:** This metric provides an average MAE across multiple folds during cross-validation, offering a more robust estimate of the model's performance.

3.4.2 Linear Regression Performance

The performance of the Linear Regression model is summarized in the table below. The Linear Regression model demonstrates relatively low loss values across the board, indicating good generalization and accurate prediction of the target variable.

Table 3.1: Linear Regression Model Losses

Training Loss	**Validation Loss**	**Test Loss (MAE)**	**Cross-Validation MAE**
0.0216	0.0375	0.0236	0.0237

The Linear Regression model shows strong performance, with a training loss of 0.0216, a validation loss of 0.0375, and a test loss (MAE) of 0.0236. This indicates that the model is not overfitting, as the difference between training and validation losses is small. Additionally, the cross-validation MAE of 0.0237 suggests consistent performance across different subsets of the data.

3.4.3 Neural Network Performance

The performance of the Neural Network model is shown below. Compared to the Linear Regression model, the Neural Network demonstrates higher losses, suggesting that it may not be as well-suited for this specific dataset or that further tuning is required.

Table 3.2: Neural Network Model Losses

Test Loss (MAE)	**Cross-Validation MAE**
0.0752	0.1431

The Neural Network model produces a higher test loss (MAE) of 0.0752 and a cross-validation MAE of 0.1431. This could indicate that the Neural Network struggles with capturing the underlying patterns in this dataset as effectively as the Linear Regression model. The higher loss values may be a result of overfitting, underfitting, or an inappropriate network structure for this task. Further tuning of hyperparameters or architectural adjustments could potentially improve its performance.

3.4.4 Why MAE is Preferred Over MSE in Regression Results

When evaluating regression models, two commonly used metrics are Mean Absolute Error (MAE) and Mean Squared Error (MSE). While both metrics provide valuable insights into a model's prediction accuracy, there are several reasons why MAE is preferred over MSE in the context of this analysis:

1. Robustness to Outliers: MSE amplifies the influence of outliers because it squares the error term. This means that large errors have a disproportionately higher impact on the overall loss. In traffic speed prediction, outliers could arise due to sudden and unrepresentative spikes or drops in speed, which might not be reflective of the general pattern. Since MAE measures the average absolute difference between predictions and actual values, it is less sensitive to these outliers, offering a more balanced measure of model performance in the presence of noisy data.

2. Interpretability: MAE is easier to interpret because it represents the average magnitude of errors in the same unit as the predicted variable. For instance, if the predicted variable is traffic speed in kilometers per hour (Km/h), the MAE directly indicates the average deviation in Km/h. MSE, on the other hand, produces squared units (Km/h^2 in this case), which can be less intuitive and harder to translate into actionable insights.

3. Linear Error Penalization: MAE penalizes errors linearly, which means that all errors contribute equally to the overall loss, regardless of their magnitude. MSE, however,

penalizes larger errors more severely due to the squared term. In scenarios where smaller, consistent deviations are acceptable but large deviations should be penalized, MSE may be more suitable. However, in traffic speed prediction, a more balanced penalization across all errors is desirable, which makes MAE a better choice.

4. Practical Relevance: In real-world applications like traffic speed prediction, a few large prediction errors (which would dominate MSE) are often less important than understanding the average error over all predictions. MAE gives a more representative measure of overall model performance, especially when the goal is to minimize the average deviation from true values without over-emphasizing outliers.

Given these factors, MAE provides a more reliable and interpretable measure of model performance for this particular task, where robustness to outliers and intuitive error interpretation are critical.[Hodson22]

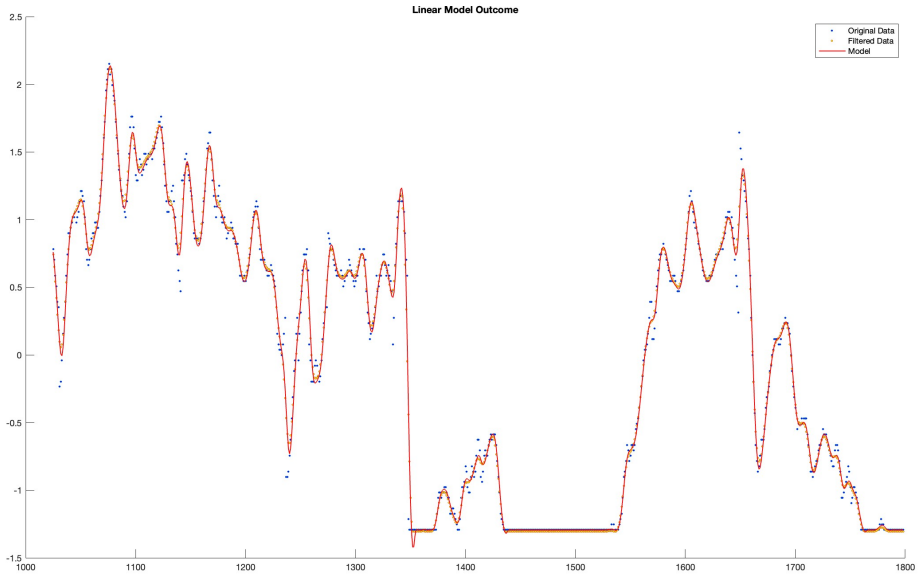
3.4.5 Regression Plots

The following figures visualize the regression outcomes for both models. These plots compare the original test data, filtered data, and the model predictions to illustrate how well each model fits the data.

The Linear Regression model's outcome is shown in Figure 3.5(a). The linear model closely follows the trends of the original data, as indicated by the red prediction line that tracks the filtered data points. This suggests that the Linear Regression model effectively captures the underlying structure of the data with minimal error.

Figure 3.6(a) illustrates the Neural Network regression model's performance. The Neural Network also captures the overall trend of the data, but it shows greater variability compared to the linear model. This corresponds to the higher loss values observed earlier. The predictions, while aligned with the overall trend, seem less accurate in regions where the data fluctuates more rapidly.

As observed, the linear model shows a closer fit to the original data with lower loss values, while the Neural Network model exhibits higher loss. Both models, however, follow the trends of the data, with the linear model performing better in terms of prediction accuracy.



(a) Linear Model Outcome

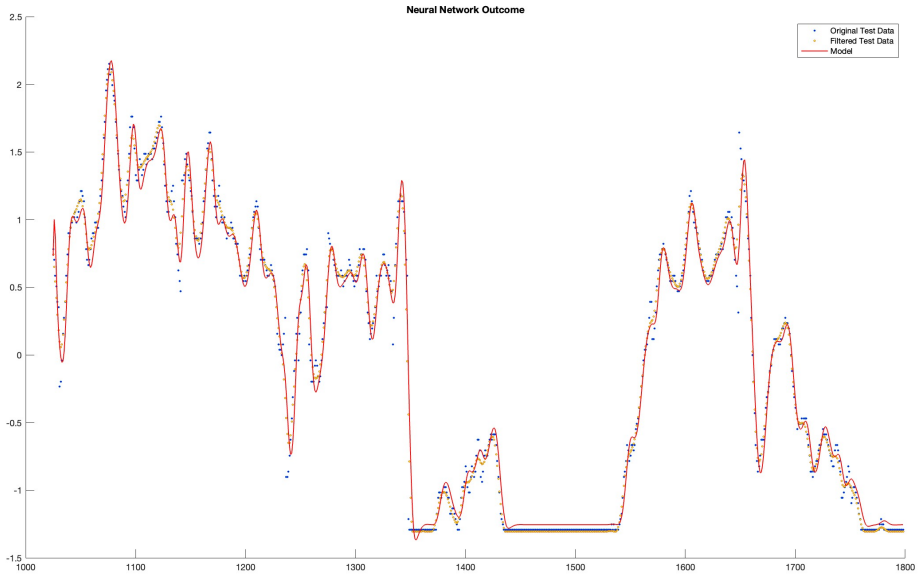
Figure 3.5: This figure compares the original traffic speed data, filtered data, and Linear Regression model predictions. The red line closely follows the filtered data, demonstrating the model's accuracy in capturing traffic trends with minimal error.

3.4.6 Discussion of Results

The comparison between the two models—Linear Regression and Neural Network—reveals significant differences in their performance when predicting traffic speed. The linear regression model consistently demonstrates lower loss values and a closer fit to the test data, as reflected in both training and validation metrics. On the other hand, the neural network, despite its potential for capturing non-linear relationships, struggles to achieve the same level of accuracy for this dataset.

The **Linear Regression model** shows strong performance, with a training loss of 0.0216, validation loss of 0.0375, and test loss (MAE) of 0.0236. The small difference between training and validation losses suggests that the model is not overfitting and generalizes well to unseen data. Moreover, the cross-validation MAE of 0.0237 confirms that the model performs consistently across different subsets of the data. These results indicate that linear regression, despite its simplicity, is well-suited for predicting traffic speed, likely due to the linear relationship between past and future traffic values.

In contrast, the **Neural Network model** exhibits higher loss values, with a test loss



(a) Neural Network Model Outcome

Figure 3.6: This figure compares the original traffic speed data, filtered data, and Neural Network predictions. The model captures the general trend but shows more variability, reflecting higher error in regions with rapid traffic fluctuations.

(MAE) of 0.0752 and a cross-validation MAE of 0.1431. These results suggest that the model may be overfitting to the training data or underperforming due to insufficient tuning of hyperparameters. While the LSTM architecture is theoretically capable of capturing complex temporal patterns, its performance here is likely hindered by the relatively small dataset and the high variability in traffic speed data. Additional tuning, such as optimizing the number of LSTM units or applying stronger regularization, could potentially improve the model's accuracy.

Both models, however, follow the general trends in the data. **Figure 3.5** shows that the linear regression model's predictions closely match the actual data, demonstrating its ability to capture the key patterns in traffic speed. In contrast, **Figure 3.6** illustrates that while the neural network captures the overall trend, its predictions are more variable, particularly in areas where the data fluctuates more rapidly.

Overall, these results suggest that for this specific dataset, **Linear Regression** offers better predictive accuracy and generalizability, while the **Neural Network** model may require further tuning to improve performance.

References

[AY20] Alqudah, N. ; Yaseen, Q.: Machine Learning for Traffic Analysis: A Review. In: *Procedia Computer Science* 170 (2020), S. 911–916

[Woo23] Woodman, R.J. ; Mangoni, A.A.: A comprehensive review of machine learning algorithms and their application in geriatric medicine: present and future. In: *Aging Clinical and Experimental Research* 35 (11) (2023), S. 2363–2397. doi: 10.1007/s40520-023-02552-2 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10627901/

[Kal23] Kalmegh, S.R. ; Padar, B.R.: Empirical Study on Evaluation Metrics for Classification Algorithms. In: *International Journal of Advanced Research in Science, Communication and Technology* 3 (2) (March 2023). https://ijarsct.co.in/Paper8898.pdf

[Gol19] Golovnin, O., Stolbova, A., and Ostroglazov, N.: An Analysis of Road Traffic Flow Characteristics Using Wavelet Transform. In: *Recent Research in Control Engineering and Decision Making*, Studies in Systems, Decision and Control, vol. 199, Springer, 2019, pp. 433–445.

[Per03] Persson, P.-O. and Strang, G.: Smoothing by Savitzky-Golay and Legendre Filters. In: *Mathematical Systems Theory in Biology, Communications, Computation, and Finance*, The IMA Volumes in Mathematics and its Applications, vol. 134, Springer, 2003, pp. 301–315.

[Shi11] Shi, Y.: The Application of the Butterworth Low-Pass Digital Filter on Experimental Data Processing. In: *2011 International Conference in Electrics, Communication and Automatic Control Proceedings*, Springer, 2011, pp. 225–230.

[Salih24] Salihoglu, S., Koksal, G., and Abar, O.: Enhancing Next Destination Prediction: A Novel LSTM Approach Using Real-World Airline Data. In: *Engineering Applications of Artificial Intelligence*, 2024. Available at: https://ar5iv.labs.arxiv.org/html/2401.12830v2.

[Deekshetha22] Deekshetha H R, A. V. Shreyas Madhav, and Amit Tyagi: Traffic Prediction Using Machine Learning. In: *Evolutionary Computing and Mobile Sustainable Networks*, pp. 969–983. Springer, 2022.

[Hodson22] Hodson, T. O.: Root-Mean-Square Error (RMSE) or Mean Absolute Error (MAE): When to Use Them or Not. *Geoscientific Model Development*, 15, 5481–5487, 2022. https://doi.org/10.5194/gmd-15-5481-2022.