RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

FUEL EFFICIENY PREDICTION

PRESENTED BY –
Allwin Joseph A (231801008)
Aadhi Raj (231801001)

AI23331 - FUNDAMENTALS OF MACHINE LEARNING

Department of Artificial Intelligence and Data Science

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME……``````````````````````……
…….

ACADEMIC YEAR……**SEMESTER**……. **BRANCH** …….

UNIVERSITY REGISTER No.

Certified that this is the Bonafide record of work done by the above students in the Mini Project titled "**TRAIN DELAY PREDICTION USING LOGISTIC REGRESSION**" in the subject **AI23331– FUNDAMENTALS OF MACHINELEARNING** during the year **2024 - 2025.**

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on ----------

Internal Examiner **External Examiner**

ABSTRACT

Abstract

This study aims to predict the fuel efficiency (measured as miles per gallon, MPG) of automobiles using a neural network-based regression model. The dataset, sourced from the UCI Machine Learning Repository, contains attributes such as engine displacement, horsepower, weight, acceleration, model year, and origin of vehicles. The data preprocessing pipeline involves handling missing values, encoding categorical features, and standardizing numerical inputs. A neural network model with two hidden layers of 64 neurons each is developed using TensorFlow and Keras. The model is trained using the Adam optimizer with Mean Squared Error as the loss function and evaluated using Mean Absolute Error and Mean Squared Error metrics. The results demonstrate the model's ability to accurately predict fuel efficiency based on the provided features, highlighting the effectiveness of neural networks in solving domain.

CHAPTER 1
INTRODUCTION

Chapter 1: Introduction - Notes

1. Background

- Fuel efficiency is a critical metric in automotive design, impacting environmental sustainability and economic factors.
- With the growing need for fuel-efficient vehicles, predicting fuel efficiency (measured as miles per gallon, MPG) has become an important task in automotive research.
- Advances in machine learning have enabled the use of predictive models to estimate MPG based on various vehicle attributes.

2. Motivation

- Accurate prediction of fuel efficiency aids manufacturers in designing optimized vehicles.
- Helps consumers make informed decisions regarding vehicle purchases.
- Addresses global concerns about energy conservation and emission reduction.

3. Problem Statement

- Traditional statistical methods may not capture complex, non-linear relationships between vehicle attributes and MPG.
- There is a need for robust machine learning models capable of handling such relationships to improve prediction accuracy.

4. Objective

- Develop a neural network model to predict the fuel efficiency (MPG) of vehicles using key attributes such as:
- Engine displacement
- Horsepower
- Weight
- Acceleration
- Model year
- Origin (categorical feature)
- Evaluate the model's performance using regression metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Objective of the study :

The objective of this study is to develop a machine learning model using TensorFlow/Keras to accurately predict vehicle fuel efficiency, measured as miles per gallon (MPG), based on key attributes such as engine displacement, horsepower, weight, acceleration, model year, and origin. Utilizing the UCI Auto MPG dataset, the study focuses on data preprocessing, including handling missing values, encoding categorical features, and scaling numerical data to enhance model performance. The neural network model is evaluated using metrics like Mean Absolute Error (MAE) and Mean Squared Error (MSE) to assess its accuracy and reliability. This study aims to provide a robust framework for predicting MPG and demonstrate the potential of machine learning in addressing challenges in automotive research, such as optimizing energy use and reducing emissions.

ALGORITHM USED

The study employs a Neural Network-based Regression Algorithm implemented using TensorFlow and Keras, designed to predict vehicle fuel efficiency (MPG) by capturing complex, non-linear relationships between features. The algorithm involves preprocessing the data by handling missing values, one-hot encoding categorical variables, and standardizing numerical features. The model architecture consists of a sequential neural network with an input layer, two hidden layers (each with 64 neurons and ReLU activation), and an output layer for continuous predictions. It uses the Adam optimizer for efficient training and Mean Squared Error (MSE) as the loss function. Model performance is evaluated using metrics such as Mean Absolute Error (MAE) and MSE to ensure accurate and reliable predictions.

CHAPTER 2 SYSTEM ARCHITECTURE

Hardware Requirements

The hardware requirements for this study, which involves training a neural network model for regression analysis, include the following:

1. Processor (CPU):
   - A multi-core processor (Intel i5 or higher, or AMD Ryzen 5 or higher) is recommended to handle the computational load during data preprocessing and model training. The more cores available, the faster the training process, especially with larger datasets.

2. Graphics Processing Unit (GPU):
   - A dedicated GPU (NVIDIA GTX 1060 or higher, or equivalent) is recommended for faster training times. Deep learning models, particularly those involving large datasets, benefit from the parallel processing capabilities of GPUs, reducing the time taken to train the model.

3. Memory (RAM):
   - At least 8GB of RAM is recommended, with 16GB or more being ideal for handling larger datasets and multiple processes simultaneously during training and evaluation.

4. Storage:
   - Minimum of 50GB of free disk space, especially for storing the dataset, model checkpoints, and training logs. SSD storage is preferred for faster data read/write operations.

5. Operating System:
   - A 64-bit operating system such as Windows 10, macOS, or a Linux distribution (e.g., Ubuntu) is required to run the software packages and libraries needed for model training.

6. Network:
   - A stable internet connection is necessary for downloading datasets, libraries, and pre-trained models if required, as well as for collaborating or sharing model results.

These hardware specifications will ensure that the study can run efficiently and handle the demands of training machine learning models, particularly deep learning networks.

CHAPTER 3 SYSTEM OVERVIEW

1. System Architecture

The system architecture for the vehicle fuel efficiency prediction model consists of several key components that work together to handle the data, model development, training, and evaluation process. The architecture is designed to ensure smooth data flow and efficient execution of the machine learning pipeline.

1. Data Collection:

- The system begins with the **data acquisition** step, where the UCI Auto MPG dataset is loaded from the internet. This dataset contains vehicle attributes and MPG values, which are essential for training the predictive model.

2. Data Preprocessing:

- Data Cleaning: The raw data is cleaned by handling missing values, removing invalid entries, and ensuring the dataset is ready for processing.
- Feature Engineering: The categorical 'Origin' feature is encoded using one-hot encoding to convert it into numerical values that can be used by the model.
- DataNormalization: The numerical features (e.g., horsepower, weight) are standardized using StandardScaler to ensure they have similar scales, which is important for the neural network training.

3. Model Development:

- Neural Network Architecture: The model is built using a Sequential Neural Network, which includes an input layer, two hidden layers (with 64 neurons each and ReLU activation), and an output layer for continuous predictions (MPG).
- Optimizer and Loss Function: The Adam optimizer is used for efficient gradient-based optimization, and Mean Squared Error (MSE) is selected as the loss function to minimize the prediction errors.

4. Training and Evaluation:

- The system trains the model using the preprocessed training data. The model learns to map the vehicle attributes to MPG using backpropagation.
- The model is evaluated on a test dataset using performance metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE) to assess its prediction accuracy.

- After training, the system is capable of making **predictions** on new or unseen data. The model predicts the MPG based on the input features of a vehicle.

Feature Engineering

Feature engineering in this study involves transforming the raw UCI Auto MPG dataset into meaningful features for model training. Missing values in the `Horsepower` column are handled by removing rows with null entries. The categorical `Origin` feature is one-hot encoded into three binary columns representing the USA, Europe, and Japan. Numerical features such as `Displacement`, `Horsepower`, `Weight`, `Acceleration`, and `Model Year` are standardized using StandardScaler to ensure consistent scaling for neural network training. Key features, including the one-hot encoded `Origin` columns, are selected based on their relevance to predicting MPG. Optional interaction features, like combinations of `Weight` and `Acceleration`, are considered for future work. These preprocessing steps ensure that the dataset is well-prepared to maximize the performance of the neural network model in predicting fuel efficiency. The logistic regression model is efficient and interpretable, making it particularly useful for applications where understanding the decision-making process is crucial. While the model may not capture highly complex non-linear relationships like some machine learning models (e.g., neural networks), it provides a transparent and reliable tool for predicting train delays. The simplicity of logistic regression allows railway operators to quickly integrate the model into existing systems and take real-time actions to address potential delays.

Conclusion

In summary, the architecture of the train delay prediction system, built around logistic regression, leverages effective data preprocessing, feature engineering, and robust evaluation techniques. By utilizing a binary classification approach, the model outputs probability scores to predict train delays based on multiple influential features. Although relatively simple compared to more complex algorithms, logistic regression remains highly effective and interpretable, offering a valuable tool for improving scheduling accuracy and operational efficiency in train management systems. The model's straightforward design also facilitates integration into real-time systems, supporting timely interventions to optimize train schedules

3.1 MODULE 1 – DATA COLLECTION AND PREPROCESSING

Data Preparation:

Data Preparation

Data preparation is a crucial step in building a machine learning model, as it ensures that the raw data is transformed into a clean and suitable format for the training process. In this study, the data preparation process for the UCI Auto MPG dataset involves several key steps:

1. Loading the Dataset:
   - The UCI Auto MPG dataset is loaded from a URL, and the appropriate column names are assigned for easy reference. The dataset contains several vehicle attributes along with the target variable, MPG (miles per gallon), which is used for prediction.

2. Handling Missing Data:
   - The dataset contains missing values, particularly in the `Horsepower` column. These missing values are handled by removing the rows that contain `NaN` values, ensuring that the model is trained on a complete dataset and eliminating the risk of introducing biases due to missing data.

3. Feature Encoding:
   - The `Origin` feature, which is categorical (with values 1, 2, 3 corresponding to USA, Europe, and Japan), is transformed into a one-hot encoded format. This process converts the categorical variable into three separate binary columns, enabling the neural network to process them as numerical inputs.

4. Feature Scaling:
   - To ensure that the model can effectively learn from the features, all numerical features such as `Displacement`, `Horsepower`, `Weight`, `Acceleration`, and `Model Year` are standardized using `StandardScaler`. This scales the features to have zero mean and unit variance, improving the model's convergence speed and stability during training.

5. Data Splitting:
   - The dataset is split into two main subsets: a **training set** and a **test set**, typically in an 80-20 or 70-30 ratio. The training set is used to train the model, while the test set is kept aside to evaluate its performance on unseen data. This helps assess the generalizability of the model.

6. Feature and Label Separation:
- The dataset is split into **features** (independent variables) and **labels** (dependent variable). The features include vehicle attributes like `Cylinders`, `Displacement`, `Horsepower`, `Weight`, `Acceleration`, `Model Year`, and the one-hot encoded `Origin` columns. The label is the `MPG` column, which the model aims to predict.

These data preparation steps ensure that the data is in the optimal format for training the neural network model, allowing it to learn the underlying patterns and relationships between the vehicle features and MPG efficiently and effectively.

Step-by-Step Algorithm for Fuel Efficiency Prediction (MPG) Using Neural Network

Here is the step-by-step process of the algorithm used to predict fuel efficiency (MPG) using a neural network, described without code:

Step 1: Load the Dataset
- Import the dataset from an external source, such as the UCI Auto MPG dataset. - Assign column names to the dataset for clarity and convenience.

Step 2: Handle Missing Data
- Identify and handle missing values within the dataset.
- Remove any rows containing missing data, especially in critical columns like `Horsepower`, to ensure that the model is not trained on incomplete information.

Step 3: Feature Encoding
- Convert categorical variables into a numerical format to allow the neural network to process them. - For the `Origin` column (which indicates the origin of the car), use one-hot encoding to create binary columns representing each category (e.g., `Origin_USA`, `Origin_Europe`, and `Origin_Japan`).

Step 4: Feature and Label Separation
- Separate the dataset into features (independent variables) and labels (dependent variable).
- The features include all the vehicle attributes, such as `Cylinders`, `Displacement`, `Horsepower`, `Weight`, `Acceleration`, and `Model Year`, along with the one-hot encoded `Origin` columns. - The label is the `MPG` column, which is the target variable the model aims to predict.

Step 5: Feature Scaling
- Standardize the numerical features to ensure they have a mean of 0 and a standard deviation of 1. - This step is important because neural networks work best when the input data is scaled, allowing the model to converge more quickly and effectively during training.

Step 6: Split the Data into Training and Testing Sets
- Split the dataset into two subsets: a training set (typically 80% of the data) and a test set (the remaining 20%).
- The training set is used to train the model, while the test set is used to evaluate its performance on unseen data.

Step 7: Define the Neural Network Model
- Define the structure of the neural network:
- An input layer that matches the number of features in the dataset.
- One or more hidden layers with neurons and activation functions (commonly ReLU) to learn the complex relationships between the input features and the output.
- An output layer with a single neuron that produces a continuous value (MPG), as this is a regression problem.

Step 8: Compile the Model
- Configure the model by selecting an optimizer (typically Adam for efficient training), a loss function (Mean Squared Error for regression), and evaluation metrics (such as Mean Absolute Error and Mean Squared Error) to monitor performance during training.

Step 9: Train the Model
- Train the neural network model using the training data.
- The model learns the relationships between the features and the target variable (MPG) by adjusting its internal weights and biases during the training process.
- The number of epochs (iterations) and batch size (number of samples per iteration) are chosen based on the training process.

Step 10: Evaluate the Model
- After training, evaluate the model on the test data to assess its generalization performance.
- The model's loss and metrics (e.g., Mean Absolute Error and Mean Squared Error) are computed to check how well it performs on unseen data.

Step 11: Make Predictions
- Once the model is trained and evaluated, use it to make predictions on new or unseen data. - The trained model predicts the MPG values based on the input features of the new dataset.

Step 12: (Optional) Visualization
- Optionally, visualize the results by plotting the loss curves during training or comparing predicted MPG values with actual MPG values from the test set.
- Visualization helps in understanding the model's learning process and performance.

Summary:
This algorithm involves preparing the dataset (loading, handling missing data, encoding features, scaling), splitting it into training and test sets, defining and training a neural network, and evaluating its performance. The model is then used to make predictions on new data to estimate vehicle fuel efficiency (MPG). The process ensures the neural network can effectively learn and generalize patterns related to fuel efficiency based on the vehicle attributes.

Results and Discussions

In this section, we present the outcomes of training the neural network model for predicting fuel efficiency (MPG) of vehicles based on various attributes like engine size, horsepower, weight, etc. We will discuss the performance metrics, insights drawn from the model's predictions, and any challenges faced during the modeling process.

Model Performance

- Training Loss: The training process of the neural network uses Mean Squared Error (MSE) as the loss function. The model begins with a relatively high loss value, but over successive epochs, the loss steadily decreases, indicating that the model is learning and optimizing its weights. This suggests that the model is effectively capturing patterns in the data.

- Test Loss and Metrics: After training, the model is evaluated on a separate test dataset. Common metrics used for evaluating regression models include:
- Mean Absolute Error (MAE):The average of the absolute differences between predicted and actual MPG values. A lower MAE indicates better prediction accuracy.
- Mean Squared Error (MSE): This metric penalizes larger errors more heavily. A low MSE value means the model is making smaller errors in predicting MPG.
- R-squared (R^2) value:Although not explicitly mentioned in the code, it's useful for evaluating the proportion of variance in the dependent variable (MPG) that is explained by the independent variables.

The results on the test set are compared to the actual values of MPG. If the MAE is sufficiently low and the R^2 value is close to 1, the model has a good predictive ability.

Model Predictions

- Predicted vs. Actual MPG:

- A plot comparing predicted MPG values with actual MPG values helps visualize how well the model is performing. Ideally, the predicted values should align closely with the actual values, demonstrating the model's accuracy.
- If there is a significant deviation between predicted and actual MPG values, it may indicate that the model is overfitting to the training data or struggling to generalize.

- Key Insights:
- The neural network model can effectively capture relationships between various vehicle attributes (e.g., `Weight`, `Horsepower`, `Acceleration`, `Cylinders`) and fuel efficiency (MPG).
- The most influential features in predicting MPG can be identified through model evaluation or feature importance analysis. For example, a vehicle's `Weight` and `Horsepower` are likely strong predictors of fuel efficiency.

Challenges and Limitations

1. Data Quality:
- The dataset had missing values, particularly in the `Horsepower` column. While removing these rows was necessary, it may have led to the loss of some potentially useful data.
- Ensuring data quality (e.g., addressing missing values or outliers) is critical for improving model performance.

2. Feature Selection:
- Some features in the dataset, like `Model Year`, `Cylinders`, and `Acceleration`, may not have been as influential in predicting MPG as others like `Weight` and `Horsepower`. Proper feature selection or dimensionality reduction could improve the model's efficiency and performance.

3. Model Complexity:
- The neural network architecture used in this study was relatively simple with just two hidden layers. While this is sufficient for this task, more complex models or techniques (e.g., deeper neural networks or tree-based methods like Random Forests) might yield better performance.
- Overfitting could be a potential issue, especially if the model is excessively complex relative to the amount of data. Regularization techniques such as dropout or L2 regularization could help mitigate overfitting.

4. Training Time:
- The model took a reasonable amount of time to train, given the size of the dataset. However, if the dataset were larger, more advanced techniques or hardware might be needed to speed up training.

Future Work and Improvements

1. **Hyperparameter Tuning:**
- Hyperparameters such as the number of hidden layers, neurons, learning rate, and batch size can significantly influence the model's performance. Grid search or random search techniques can be applied to find the optimal hyperparameters.

2. **Advanced Models:**
- Exploring more advanced machine learning models like Random Forests, Gradient Boosting, or Support Vector Machines (SVM) could potentially provide better predictive accuracy for fuel efficiency.
- Deep learning techniques, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), could also be explored, although they might be more suited for image or time-series data, respectively.

3. **Feature Engineering:**
- More advanced feature engineering techniques could be explored. For example, interactions between features (e.g., `Weight` * `Horsepower`) could be tested to see if they improve the model's predictive power.

- Including additional external data such as car brand, fuel type, or even geographic location of vehicles could further enhance the model's accuracy.

Conclusion

The neural network model for predicting vehicle fuel efficiency (MPG) demonstrated solid performance with relatively low error metrics. The model was able to effectively use the vehicle attributes to make accurate predictions, though there is room for improvement. Addressing challenges such as feature selection, hyperparameter tuning, and potential overfitting will further enhance the model's predictive power and robustness. Exploring more advanced models and feature engineering techniques could also help improve the accuracy and generalization of the predictions.

Source code

```
import pandas as pd import numpy as np import
matplotlib.pyplot as plt from
sklearn.model_selection import train_test_split from
sklearn.preprocessing import StandardScaler import
tensorflow as tf
from tensorflow.keras import layers

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
column_names = ['MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight', 'Acceleration', 'Model Year',
'Origin']
data = pd.read_csv(url, names=column_names, na_values="?", comment='\t', sep=" ",
skipinitialspace=True)

data = data.dropna()

data['Origin'] = data['Origin'].map({1: 'USA', 2: 'Europe', 3: 'Japan'}) data
= pd.get_dummies(data, columns=['Origin'], prefix='', prefix_sep='')

features = data.drop('MPG', axis=1) labels
= data['MPG']

scaler = StandardScaler()
features = scaler.fit_transform(features)

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)

model = tf.keras
```

CONCLUSION:

The above report detailed the algorithms and importance of fuel prediction in todays world.