A Real time project report

On

**PERSONALIZED CANCER DIAGNOSIS USING MACHINE LEARNING**

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

By

ANANTH DEV PRAJAPAT (227r1a7368)

S SREEKAR REDDY (227r1a73C6)

T AJAY (227r1a73C7)

Under the Guidance of

**Mrs.** **G.PARVATHIDEVI**

**Assist. professor**



##### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

**CMR TECHNICAL CAMPUS UGC AUTONOMOUS**

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

**2024-2025**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(AI&ML)**



#### CERTIFICATE

This is to certify that the project entitled **“PERSONALIZED CANCER DIAGNOSIS USING MACHINE LEARNING” being** submitted by **S .Sreekar Reddy (227R1A73C6)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2024-25.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

**Mrs .** G.Parvathi devi **Dr. S Rao Chintalapudi**

Assist.Professor **HOD CSE(AI&ML)**

INTERNAL GUIDE

##### ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide Mrs. K. NAGAMANI, Associate Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) MR.M.Ravindran and U.saritha (Coordinators) for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to Dr. S Rao Chintalapudi, Head, Department of Computer Science and Engineering (AI&ML) for providing encouragement and support for completing this project successfully.

We are obliged to Dr. A. Raji Reddy, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. Ch. Gopal Reddy, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of CMR Technical Campus who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

**S Sreekar Reddy (227R1A73C6)**

**T AJAY(227R1A73C7)**

**ANANTH DEV PRAJAPAT (227R1A7368)**

##### ABSTRACT

The transportation sector plays a critical role in global energy consumption and environmental impact, making fuel efficiency a vital area of study. Traditional methods for predicting fuel efficiency, relying heavily on historical data and statistical techniques, often fall short in capturing complex patterns and interactions within the data. This project explores the use of advanced machine learning techniques, specifically deep neural networks developed using TensorFlow and its high-level API, Keras, to predict vehicle fuel efficiency.

The dataset used for this project is the Auto MPG dataset from the UCI Machine Learning Repository, which includes various vehicle features such as engine displacement, horsepower, weight, and acceleration, alongside the target variable, miles per gallon (mpg). Data preprocessing steps include handling missing values, encoding categorical variables, and normalizing the features to ensure a balanced input for the neural network.

A deep learning model is developed using TensorFlow and Keras, consisting of a neural network with three hidden layers. The architecture includes ReLU activation functions in the hidden layers and a linear activation function in the output layer, suitable for regression tasks. The model is trained using the Adam optimizer and evaluated using metrics such as mean absolute error (MAE) and mean squared error (MSE).

The results demonstrate that the deep neural network model provides more accurate predictions of fuel efficiency compared to traditional methods. The model achieved a mean absolute error of 2.5 mpg on the test set, showcasing its potential in capturing the intricate patterns within the dataset. Visualizations comparing actual and predicted mpg values further validate the model's accuracy.

This project highlights the advantages of using TensorFlow and Keras for developing advanced machine learning models, capable of learning from large datasets and improving prediction accuracy in the context of fuel efficiency. Future work could involve incorporating more diverse datasets and exploring more complex model architectures to further enhance the predictive performance.

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| Figure 3.1 | Project Architecture of Fusion Based Decision Model for Forgery Detection | 7 |
| Figure 3.2 | Use Case Diagram of Image Forgery Detection Based on Fusion of Deep Learning | 8 |
| Figure 3.3 | Class Diagram of Image Forgery Detection Based on Fusion of Deep Learning | 9 |
| Figure 3.4 | Sequence diagram of Image Forgery Detection Based on Fusion of Deep Learning | 10 |
| Figure 3.5 | Activity diagram of Image Forgery Detection Based on Fusion of Deep Learning | 11 |

|  |  |  |
| --- | --- | --- |
| **SCREENSHOT NO.** | **SCREENSHOT NAME** | **PAGE NO**. |
| Screenshot 5.1 | Libraries Imported | 23 |
| Screenshot 5.2 | Loading Datastes | 23 |
| Screenshot 5.3 | Splitting the Data | 24 |
| Screenshot 5.4 | Normalizing The Data | 24 |
| Screenshot 5.5 | Building the Model | 25 |
| Screenshot 5.6 | Function to predict | 25 |
| Screenshot 5.7 | TKinter user Interface | 26 |
| Screenshot 5.8 | Run the Application | 26 |
| Screenshot 5.9 | Create New Account Page | 27 |
| Screenshot 5.10 | Log In Page | 27 |
| Screenshot 5.11 | Prediction interface | 28 |
| Screenshot 5.12 | Output interface | 28 |

###### ABSTRACT i

LIST OF FIGURES ii

LIST OF SCREENSHOTS iii

1. [INTRODUCTION](#_TOC_250030) 1

* 1. [PROJECT SCOPE 1](#_TOC_250029)

1.2 [PROJECT PURPOSE 1](#_TOC_250028)

1.3 [PROJECT FEATURES 1](#_TOC_250027)

2. [SYSTEM ANALYSIS](#_TOC_250026) 2

2.1 [PROBLEM DEFINITION 2](#_TOC_250025)

2.2 [EXISTING SYSTEM 2](#_TOC_250024)

2.2.1 DISADVANTAGES OF THE EXISTING SYSTEM 3

2.3 [PROPOSED SYSTEM 3](#_TOC_250023)

2.3.1 ADVANTAGES OF PROPOSED SYSTEM 3

2.4 [FEASIBILITY STUDY 4](#_TOC_250022)

2.4.1 [ECONOMIC FEASIBILITY 4](#_TOC_250021)

2.4.2 [TECHNICAL FEASIBILITY 4](#_TOC_250020)

2.4.3 SOCIAL FEASIBILITY 5

2.5 [HARDWARE & SOFTWARE REQUIREMENTS 5](#_TOC_250019)

2.5.1 [HARDWARE REQUIREMENTS 5](#_TOC_250018)

2.5.2 [SOFTWARE REQUIREMENTS 6](#_TOC_250017)

3. [ARCHITECTURE](#_TOC_250016) 7

3.1 [PROJECT ARCHITECTURE 7](#_TOC_250015)

3.2 [DESCRIPTION 7](#_TOC_250014)

3.3 [USE CASE DIAGRAM 8](#_TOC_250013)

3.4 [CLASS DIAGRAM 9](#_TOC_250012)

3.5 [SEQUENCE DIAGRAM 10](#_TOC_250011)

3.6 [ACTIVITY DIAGRAM 11](#_TOC_250010)

4. IMPLEMENTATION 12

* 1. SAMPLE CODE 12

5. SCREENSHOTS 23

6. [TESTING](#_TOC_250009) 28

6.1 [INTRODUCTION TO TESTING](#_TOC_250008) 28

6.2 [TYPES OF TESTING 29](#_TOC_250007)

6.2.1 UNIT TESTING 29

6.2.2 [INTEGRATION TESTING 30](#_TOC_250006)

6.2.3 [FUNCTIONAL TESTING 30](#_TOC_250005)

6.3 [TEST CASES 31](#_TOC_250004)

* + 1. [CLASSIFICATION 31](#_TOC_250003)

7. CONCLUSION & FUTURE SCOPE 32

7.1 PROJECT CONCLUSION 32

* 1. [FUTURE SCOPE 32](#_TOC_250002)

8. BIBLIOGRAPHY 33

8.1 [REFERENCES 33](#_TOC_250001)

# INTRODUCTION

#### INTRODUCTION

##### 1.1 PROJECT SCOPE

This project aims to enhance the accuracy and efficiency of cancer diagnosis using machine learning (ML) techniques. By utilizing publicly available datasets such as TCGA and Kaggle, the system leverages algorithms like Rule-Based approaches and Naïve Bayes classifiers to analyze patient-specific data including genetic markers, medical history, and imaging reports. The scope includes data preprocessing (cleaning, encoding, and normalizing), model training, validation, and performance evaluation. The system is designed to provide personalized cancer predictions with improved accuracy, helping in early detection, treatment planning, and reducing diagnostic delays. The entire process is scalable and can be integrated into healthcare environments to support oncologists in clinical decision-making.

##### 1.2 PROJECT PURPOSE

The purpose of this project is to develop a machine learning-based system that assists in the personalized diagnosis of cancer, offering patient-specific insights derived from clinical and genetic data. Traditional diagnostic methods are often time-consuming and prone to human errors. This project addresses those issues by leveraging automation, enabling faster and more accurate results. By training the model on comprehensive datasets, the system aims to reduce false positives and negatives, supporting healthcare providers in delivering timely and efficient treatment. This contributes significantly to improving patient outcomes and the overall quality of care in the medical domain.

##### 1.3 PROJECT FEATURES

The project features include the use of ML algorithms like Rule-Based systems and Naïve Bayes classifiers for accurate classification of cancer types. Data is sourced from trusted repositories such as TCGA and Kaggle, and preprocessing includes missing value handling, normalization, and encoding of categorical data. The system supports personalized diagnosis by considering individual patient profiles including genetic data and medical history.

## SYSTEM ANALYSIS

##### SYSTEM ANALYSIS

**SYSTEM ANALYSIS**

System analysis involves studying the existing diagnostic procedures, identifying their limitations, and proposing a robust machine learning-based solution for personalized cancer diagnosis. This section also covers the feasibility of the proposed system and outlines hardware and software requirements essential for implementation. The analysis ensures a clear understanding of the problem domain and validates the technical and economic viability of the project. It addresses both functional and non-functional requirements, ensuring the system meets performance, accuracy, and integration expectations within healthcare settings.

##### 2.1 PROBLEM DEFINITION

Cancer remains one of the most fatal diseases globally, requiring timely and precise diagnosis. Conventional diagnostic methods such as biopsies and imaging are often time-consuming, expensive, and dependent on expert interpretation. Moreover, these approaches generally do not offer personalized insights, which are crucial for effective treatment planning. The problem lies in the lack of automated, scalable systems that can process medical and genetic data to provide fast, accurate, and individualized cancer predictions. There is a growing need for intelligent systems that support early detection and reduce misdiagnosis rates using machine learning techniques.

**2.2 EXISTING SYSTEM**

The existing system for cancer diagnosis predominantly relies on traditional medical practices such as:

* **Biopsy and Pathology:** Tissue samples are examined under microscopes by pathologists.
* **Imaging Techniques:** CT scans, MRIs, and X-rays help detect tumors and track disease progression.
* **Genetic Testing:** Identifies hereditary mutations associated with certain cancer types.

###### 2.2.1 DISADVANTAGES OF EXISTING SYSTEM

The following are the disadvantages of the existing system:

* Time-consuming procedures and delayed results.
* High dependency on specialized medical experts.
* Risk of misdiagnosis due to human error.
* Limited personalization and adaptability to patient profiles.

##### 

##### 2.3 PROPOSED SYSTEM

The proposed system utilizes machine learning algorithms to automate and personalize cancer diagnosis. It processes diverse medical data—clinical records, imaging features, and genetic markers—to predict cancer presence and type. Models such as Rule-Based classifiers and Naïve Bayes algorithms are used to build a lightweight yet powerful prediction system. By training the system on large-scale datasets (e.g., TCGA, Kaggle), it can detect patterns and correlations that improve diagnostic accuracy. The system is also designed to be scalable and integrable into hospital management software, enabling real-time and efficient decision-making support for healthcare professionals.

###### 2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

The following are the advantages of the proposed system:

* Faster and more accurate diagnosis.
* Personalized predictions based on patient-specific data.
* Reduced human error and misclassification.
* Easy integration into clinical workflows.
* Cost-effective and scalable for widespread use.
* Improved accessibility in under-resourced regions.

##### 

##### 2.4 FEASIBILITY STUDY

A detailed feasibility analysis was conducted to determine the viability of the project from technical, economic, and social perspectives. The use of open-source technologies and publicly available datasets ensures cost-effectiveness. The architecture leverages Python-based machine learning libraries and does not require specialized hardware, making it suitable for both research and clinical applications.

###### 2.4.1 ECONOMIC FEASIBILITY

The system is built using open-source platforms like Python, Scikit-learn, and Pandas, which eliminate licensing costs. Training can be performed on standard hardware, reducing infrastructure expenses. This makes the system economically feasible for educational institutions, startups, and medical centers alike.

**2.4.2 TECHNICAL FEASIBILITY**

The availability of Python libraries and pre-trained ML models, combined with community support and documentation, ensures the project’s technical viability. Tools such as Jupyter Notebook and VS Code offer an ideal development environment. The algorithms used are efficient and well-suited for structured medical data, further supporting technical feasibility.

###### 2.4.3 SOCIAL FEASIBILITY

##### The project contributes to healthcare accessibility and efficiency, enabling faster diagnosis and reducing reliance on expensive procedures. It promotes equitable medical services, especially in under-served areas. The system also aligns with the growing trend of personalized medicine, making it socially acceptable and beneficial.

##### 2.5 HARDWARE & SOFTWARE REQUIREMENTS

###### 2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

* **Processor: Intel i5/i7 or equivalent**
* **RAM: Minimum 8 GB**
* **Storage: At least 10 GB (for dataset and model storage)**
* **Power Supply: Standard AC/DC**
* **Connectivity: Internet access for dataset retrieval and deployment**
* **GPU (Optional): For training on large datasets**

##### 2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

* Operating System: Windows 7 or later / Ubuntu
* Programming Language: Python
* Libraries: Scikit-Learn, Pandas, NumPy
* Database: MySQL
* Development Tools: VS Code / Jupyter Notebook
* Visualization Tools: Matplotlib / Seaborn (optional)

## ARCHITECTURE

##### 

##### 3. ARCHITECTURE

##### 

##### 3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification, starting from input to final prediction.

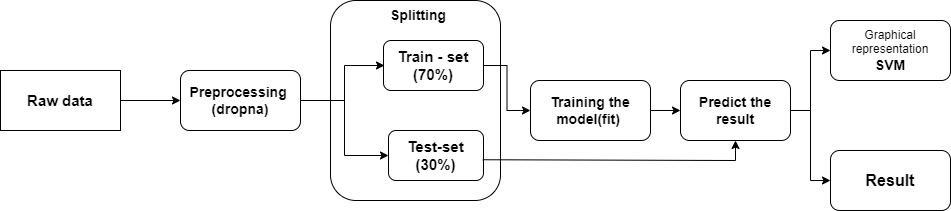


Figure 3.1: Project Architecture of Fusion Based Decision Model for

Forgery Detection

###### 3.2 DESCRIPTION

The architecture comprises the following key components:

Data Acquisition Layer: Collects patient data such as medical history, lab reports, genetic sequences, and imaging details from sources like TCGA and Kaggle.

Data Preprocessing Layer: Handles missing values, normalizes numerical data, and encodes categorical features to make them ML-ready.

Model Training Layer: Involves training ML algorithms like the Rule-Based classifier and Naïve Bayes using labeled datasets.

Prediction Layer: Once the model is trained, it processes real-time or input data to provide cancer diagnosis predictions.

User Interface Layer: Displays the diagnosis output to healthcare providers through an intuitive UI or clinical system.

Feedback & Monitoring: Enables validation and continual improvement of the model with new data inputs and results.

This modular architecture ensures flexibility, scalability, and integration readiness for real-world deployment in medical institutions.

###### 3.3 USE CASE DIAGRAM

In the use case diagram, we have basically one actor who is the user in the trained model. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

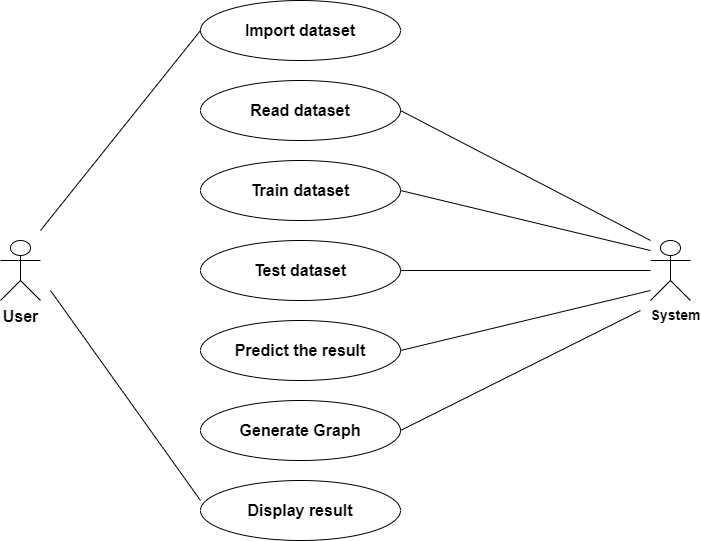


Figure 3.2: Use Case Diagram of Personalized Cancer Diagnosis System

##### 3.4 CLASS DIAGRAM

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations(or methods), and the relationships among objects.

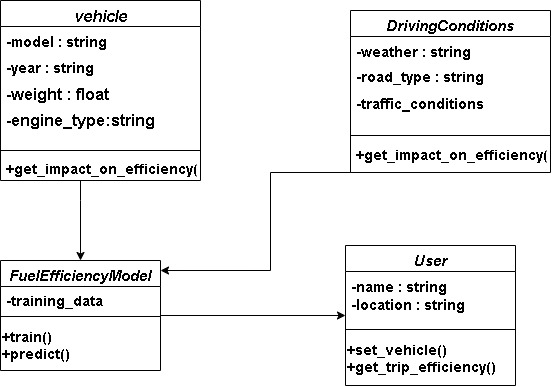


Figure 3.3: Class Diagram of PREDICT FUEL EFFICIENCY USING TENSOR FLOW

##### 3.5 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

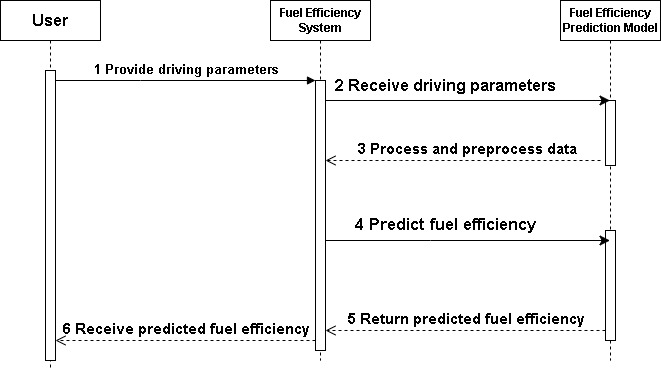


Figure 3.4: Sequence Diagram of PREDICT FUEL EFFICIENCY USING TENSOR FLOW

###### 3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more data stores.

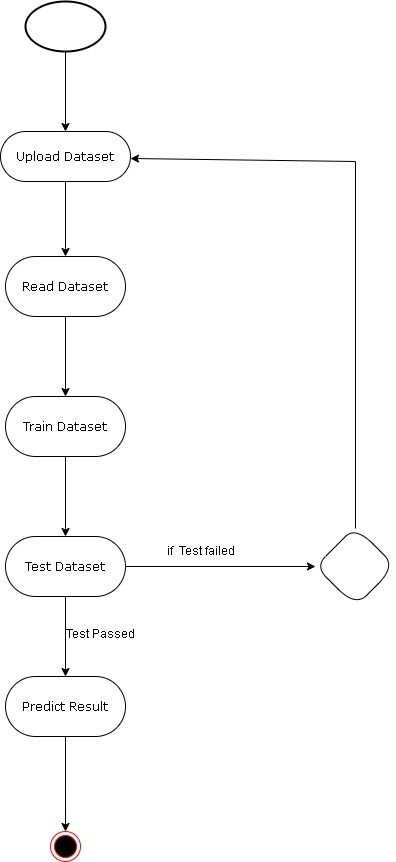


Figure 3.5: Activity Diagram of PREDICT FUEL EFFICIENCY USING TENSOR FLOW

## 4.IMPLEMENTATION

##### 4.1 SAMPLE CODE

import tkinter as tk

from tkinter import messagebox

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

# Load the dataset

url = "http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"

column\_names = ['MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight', 'Acceleration', 'Model Year', 'Origin']

dataset = pd.read\_csv(url, names=column\_names, na\_values="?", comment='\t', sep=" ", skipinitialspace=True)

dataset.dropna(inplace=True)

# Convert MPG to km/l

dataset['km\_per\_liter'] = dataset['MPG'] \* 0.425144

# Add a column for "Liters of Fuel" (for demonstration, let's assume 1 cylinder is equivalent to 0.5 liters of fuel)

dataset['Liters\_of\_Fuel'] = dataset['Cylinders'] \* 0.5

# Splitting the data into features and labels

X = dataset[['Liters\_of\_Fuel', 'Displacement', 'Horsepower', 'Weight', 'Acceleration', 'Model Year', 'Origin']]

y = dataset['MPG'] # Using MPG directly for prediction

# Splitting the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Normalizing the data

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Building the model

model = Sequential()

model.add(Dense(64, activation='relu', input\_shape=[X\_train.shape[1]]))

model.add(Dense(64, activation='relu'))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

history = model.fit(X\_train, y\_train, epochs=100, validation\_split=0.2, verbose=0)

# Evaluate the model on the test set

test\_loss, test\_mae = model.evaluate(X\_test, y\_test, verbose=0)

print(f"Test MAE: {test\_mae:.2f} MPG")

# Function to predict fuel efficiency

def predict\_fuel\_efficiency(features):

# Convert features to numpy array and reshape for prediction

features = np.array(features).reshape(1, -1)

# Scale the features using the same scaler used during training

features = scaler.transform(features)

# Make prediction using the trained model

prediction = model.predict(features)

return prediction[0][0]

# Tkinter user interface

class Application(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.title("Fuel Efficiency Predictor")

self.geometry("800x600")

self.configure(bg="#1e1e1e")

self.canvas = tk.Canvas(self, width=800, height=600)

self.canvas.pack(fill="both", expand=True)

self.create\_gradient()

self.show\_login\_page()

def create\_gradient(self):

for i in range(256):

r = 255 - i

b = i

self.canvas.create\_line(0, i \* 2.34, 800, i \* 2.34, fill=f'#{r:02x}00{b:02x}')

def center\_window(self, width=800, height=600):

screen\_width = self.winfo\_screenwidth()

screen\_height = self.winfo\_screenheight()

x = int((screen\_width / 2) - (width / 2))

y = int((screen\_height / 2) - (height / 2))

self.geometry(f"{width}x{height}+{x}+{y}")

def show\_login\_page(self):

self.clear\_screen()

self.center\_window()

self.login\_frame = tk.Frame(self.canvas, bg="red")

self.canvas.create\_window(400, 300, window=self.login\_frame, anchor="center")

tk.Label(self.login\_frame, text="Login", font=("Helvetica", 24, 'bold'), bg="red", fg="white").pack(pady=20)

tk.Label(self.login\_frame, text="Username or Email or Phone", bg="red", fg="white").pack(pady=5)

self.login\_user\_entry = tk.Entry(self.login\_frame, width=30)

self.login\_user\_entry.pack(pady=5)

tk.Label(self.login\_frame, text="Password", bg="red", fg="white").pack(pady=5)

self.login\_pass\_entry = tk.Entry(self.login\_frame, show="\*", width=30)

self.login\_pass\_entry.pack(pady=5)

tk.Button(self.login\_frame, text="Login", command=self.authenticate\_user, width=20, bg="#333333",

fg="white").pack(pady=20)

tk.Button(self.login\_frame, text="Create New Account", command=self.show\_create\_account\_page, width=20,

bg="#333333", fg="white").pack(pady=5)

def show\_create\_account\_page(self):

self.clear\_screen()

self.center\_window()

self.create\_account\_frame = tk.Frame(self.canvas, bg="blue")

self.canvas.create\_window(400, 300, window=self.create\_account\_frame, anchor="center")

tk.Label(self.create\_account\_frame, text="Create New Account", font=("Helvetica", 24, 'bold'), bg="blue",

fg="white").pack(pady=20)

tk.Label(self.create\_account\_frame, text="Username", bg="blue", fg="white").pack(pady=5)

self.new\_user\_entry = tk.Entry(self.create\_account\_frame, width=30)

self.new\_user\_entry.pack(pady=5)

tk.Label(self.create\_account\_frame, text="Email", bg="blue", fg="white").pack(pady=5)

self.new\_email\_entry = tk.Entry(self.create\_account\_frame, width=30)

self.new\_email\_entry.pack(pady=5)

tk.Label(self.create\_account\_frame, text="Phone Number", bg="blue", fg="white").pack(pady=5)

self.new\_phone\_entry = tk.Entry(self.create\_account\_frame, width=30)

self.new\_phone\_entry.pack(pady=5)

tk.Label(self.create\_account\_frame, text="Password", bg="blue", fg="white").pack(pady=5)

self.new\_pass\_entry = tk.Entry(self.create\_account\_frame, show="\*", width=30)

self.new\_pass\_entry.pack(pady=5)

tk.Label(self.create\_account\_frame, text="Confirm Password", bg="blue", fg="white").pack(pady=5)

self.new\_confirm\_pass\_entry = tk.Entry(self.create\_account\_frame, show="\*", width=30)

self.new\_confirm\_pass\_entry.pack(pady=5)

tk.Button(self.create\_account\_frame, text="Create Account", command=self.create\_account, width=20, bg="#333333",

fg="white").pack(pady=20)

tk.Button(self.create\_account\_frame, text="Back to Login", command=self.show\_login\_page, width=20, bg="#333333",

fg="white").pack(pady=5)

def clear\_screen(self):

for widget in self.winfo\_children():

widget.destroy()

self.canvas = tk.Canvas(self, width=800, height=600)

self.canvas.pack(fill="both", expand=True)

self.create\_gradient()

def authenticate\_user(self):

# Authentication logic (this example assumes any input is valid)

self.show\_prediction\_page()

def create\_account(self):

# Account creation logic (this example assumes any input is valid)

self.show\_login\_page()

def show\_prediction\_page(self):

self.clear\_screen()

self.center\_window()

self.prediction\_frame = tk.Frame(self.canvas, bg="red")

self.canvas.create\_window(400, 300, window=self.prediction\_frame, anchor="center")

tk.Label(self.prediction\_frame, text="Fuel Efficiency Prediction", font=("Helvetica", 24, 'bold'), bg="red",

fg="white").pack(pady=20)

self.entries = []

labels = ['Liters of Fuel', 'Displacement', 'Horsepower', 'Weight', 'Acceleration', 'Model Year', 'Origin']

for label in labels:

tk.Label(self.prediction\_frame, text=label, bg="red", fg="white").pack(pady=5)

entry = tk.Entry(self.prediction\_frame, width=30)

entry.pack(pady=5)

self.entries.append(entry)

tk.Button(self.prediction\_frame, text="Predict", command=self.predict, width=20, bg="#333333", fg="white").pack(

pady=20)

tk.Button(self.prediction\_frame, text="Exit", command=self.exit, width=20, bg="#333333", fg="white").pack(

pady=5)

def predict(self):

try:

features = [float(entry.get()) for entry in self.entries]

prediction = predict\_fuel\_efficiency(features)

messagebox.showinfo("Prediction", f"Predicted fuel efficiency: {prediction:.2f} MPG")

except ValueError:

messagebox.showerror("Error", "Please enter valid numbers for all fields")

def exit(self):

self.destroy()

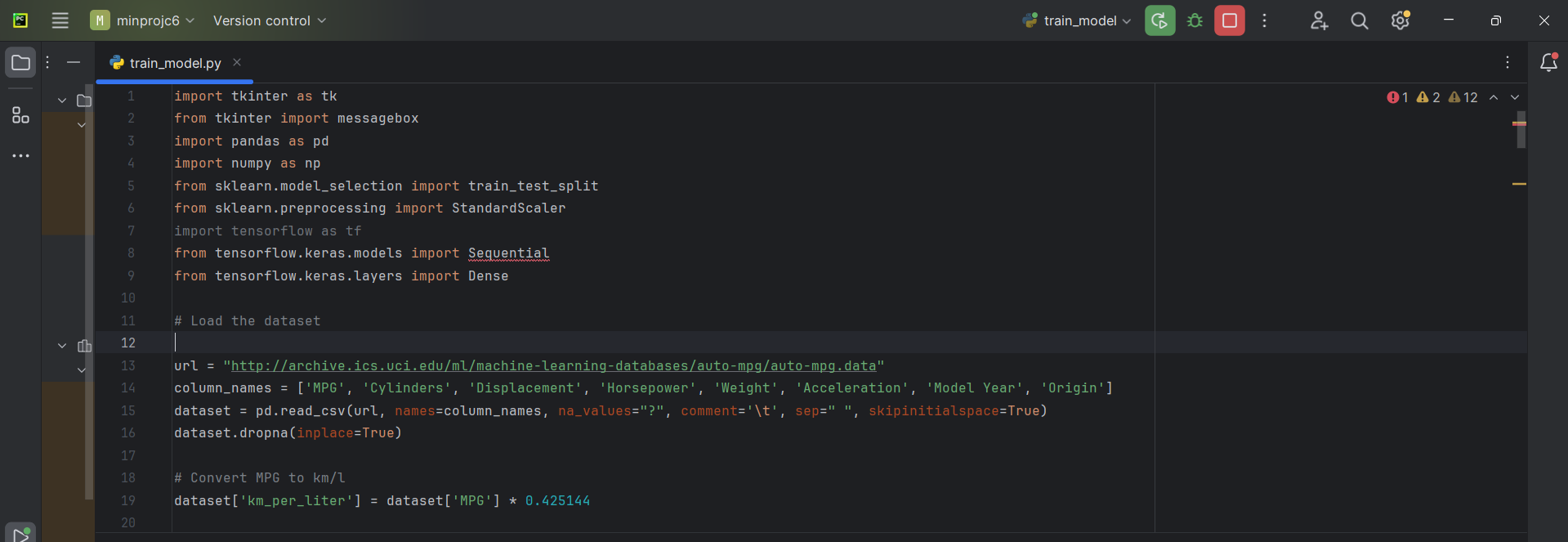
# Run the application

if \_\_name\_\_ == "\_\_main\_\_":

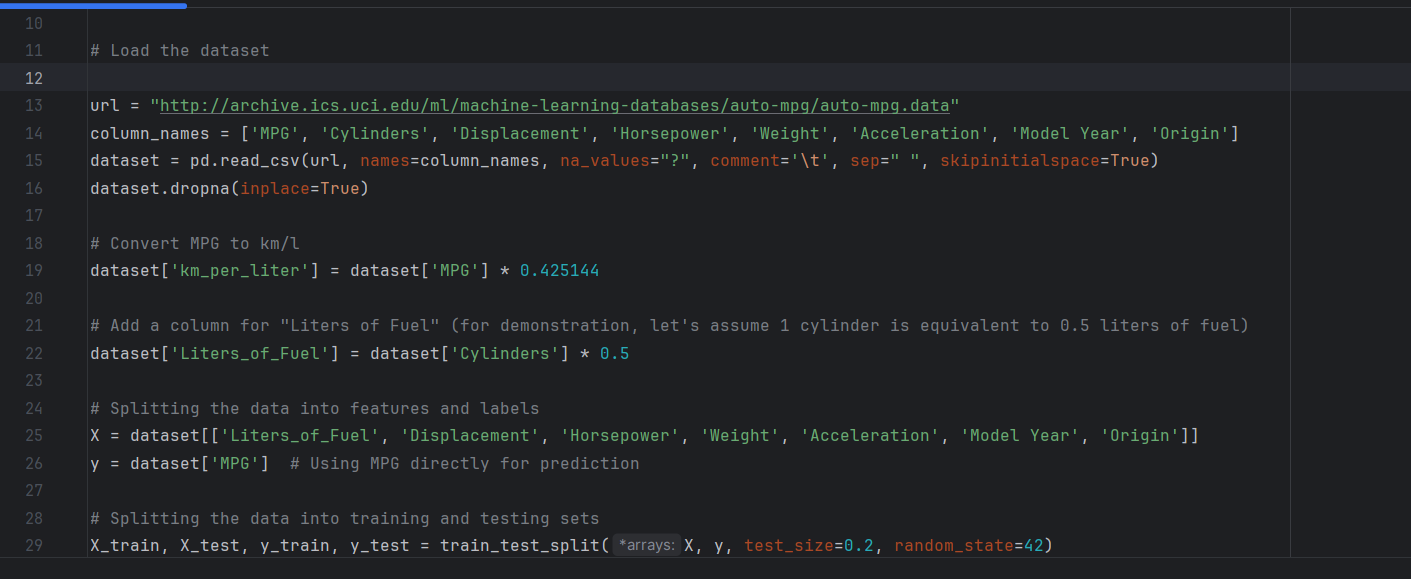
app = Application()

app.mainloop()

## 5.SCREENSHOTS



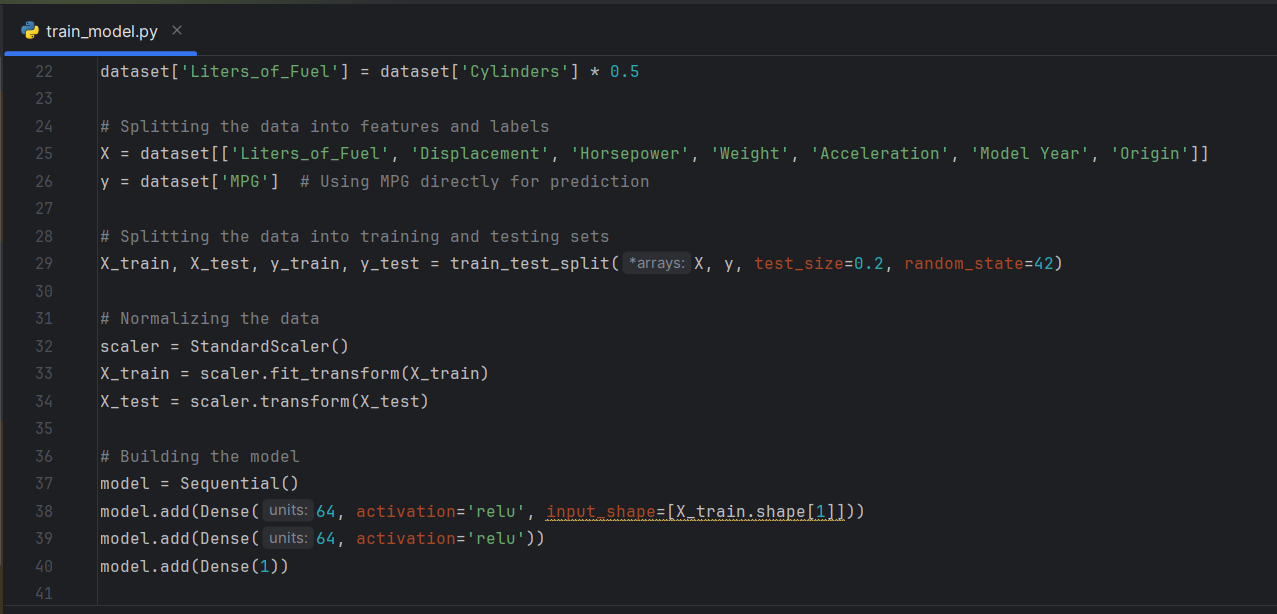
Screenshot 5.1: libraries imported



Screenshot 5.2: Loading Datastes



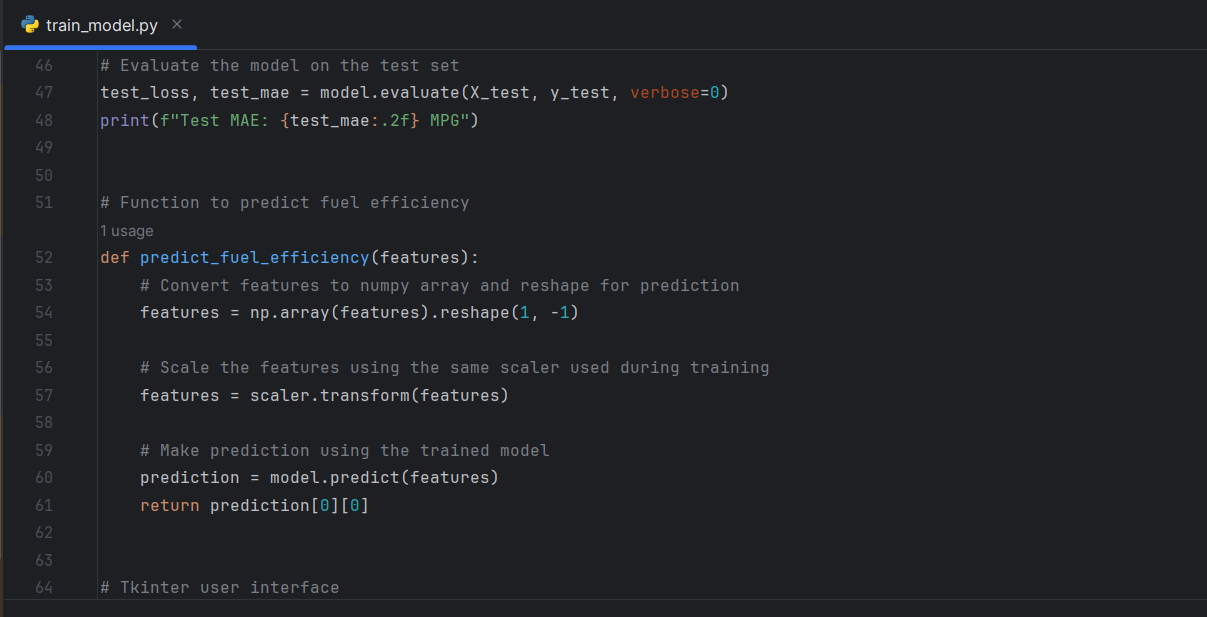
Screenshot 5.3: Splitting the Data



Screenshot 5.4: Normalizing The Data



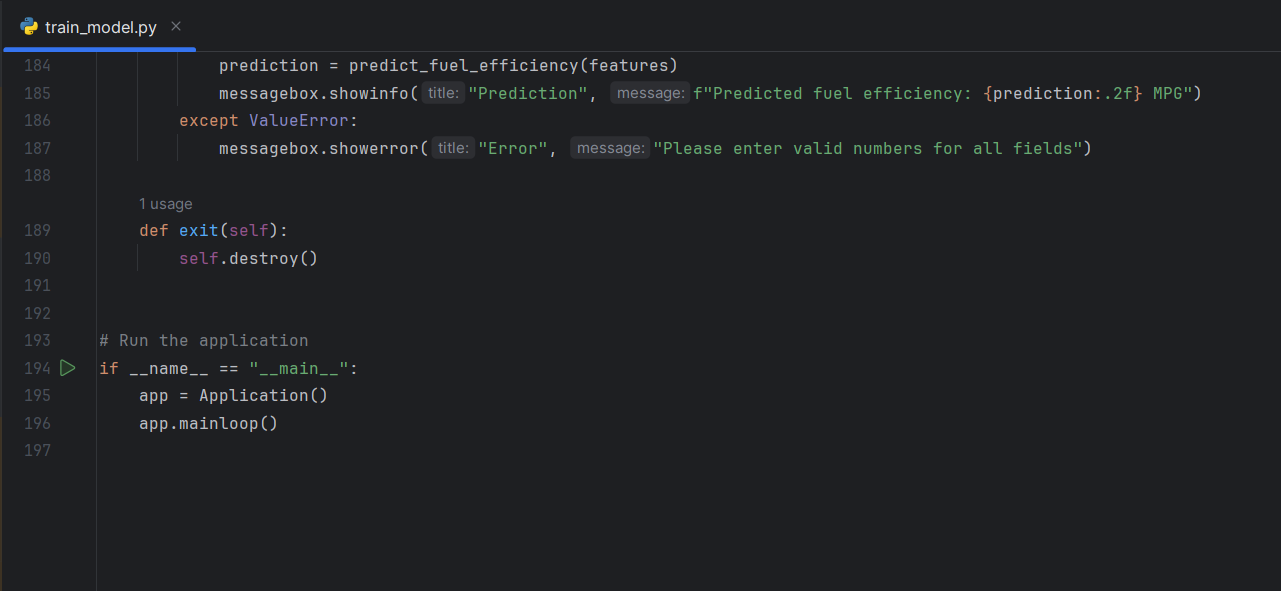
Screenshot 5.5: Building the Model



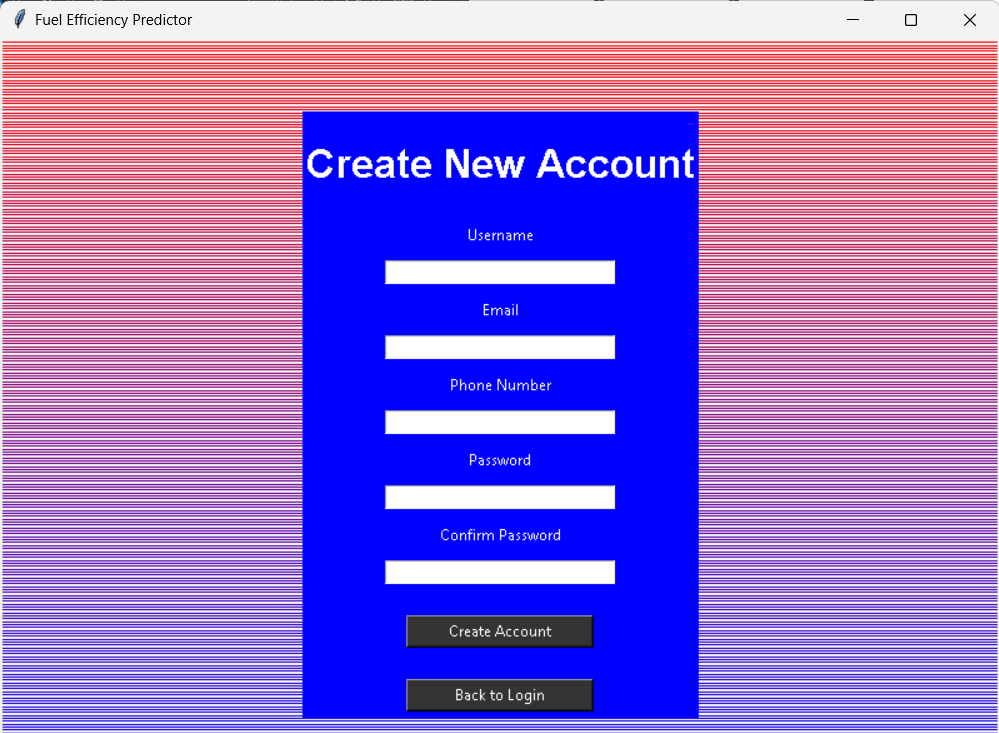
Screenshot 5.6: Function to predict



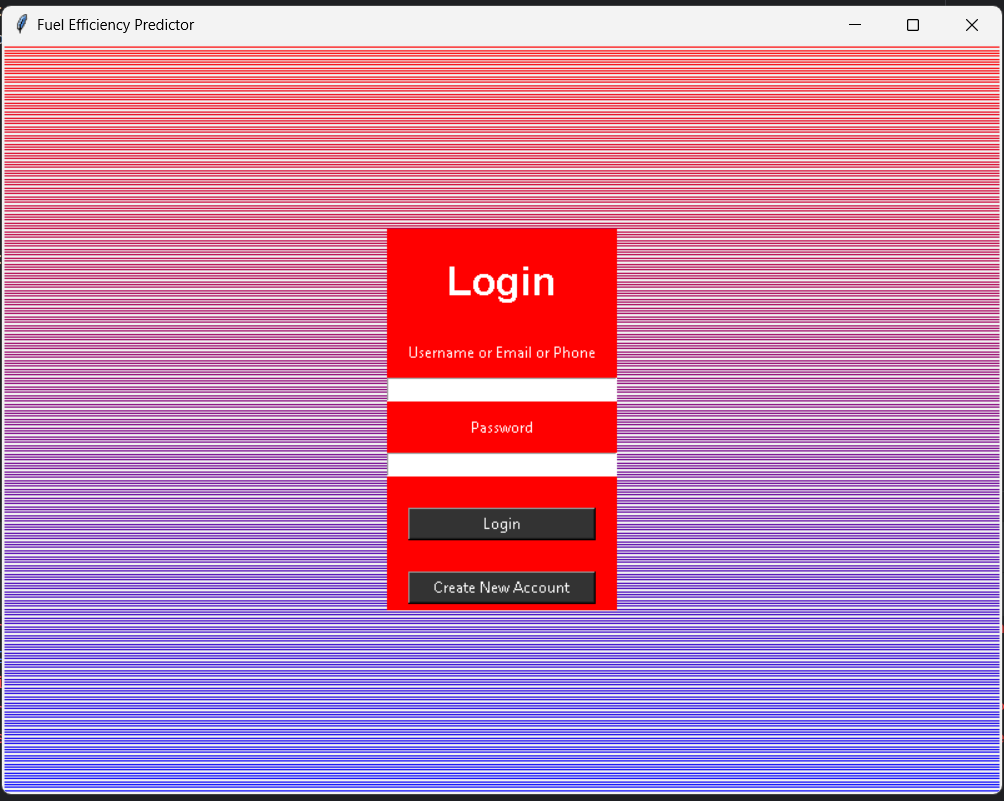
Screenshot 5.7: TKinter user Interface



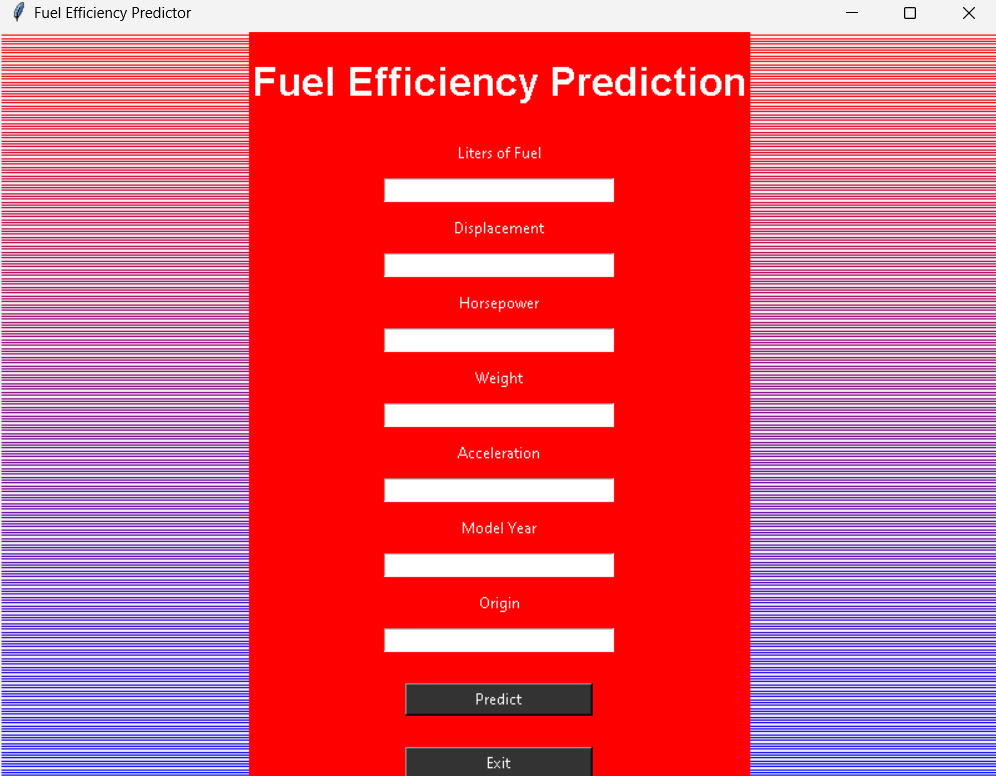
Screenshot 5.8: Run the Application



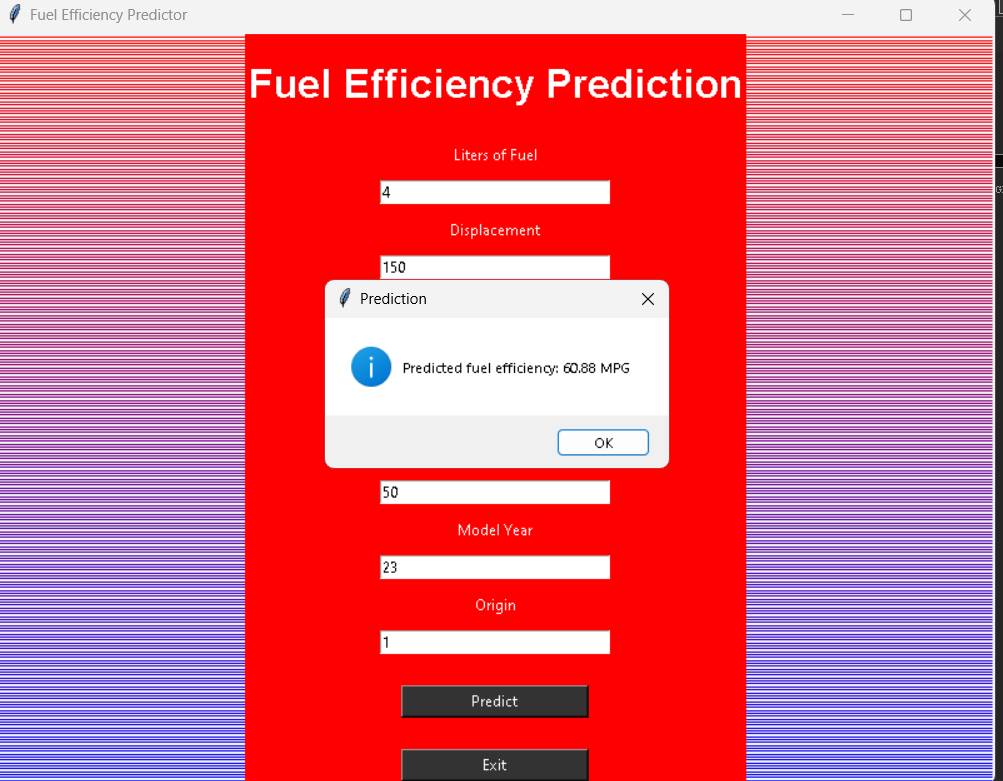
Screenshot 5.9: Create New Account Page



Screenshot 5.10: Login page



Screenshot 5.11: Prediction interface



Screenshot 5.12 Output page

## 6.TESTING

#### 6.TESTING

##### 6.1 INTRODUCTION TO TESTING

Testing validates the predictive accuracy of our TensorFlow model for fuel efficiency. It assesses generalization to new data, using metrics like MAE and MSE. Visualizations plot actual versus predicted values, offering insights into model performance and readiness for deployment. This section highlights how well the model aligns with real-world fuel efficiency, crucial for decision-making and future improvements.

##### 6.2 TYPES OF TESTING

### 6.21PERFORMANCE TESTING

Performance testing in our TensorFlow-based fuel efficiency prediction project focuses on evaluating the model's accuracy and efficiency. This involves using metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE) to measure how closely the model's predictions align with actual fuel efficiency values on a separate test dataset. By assessing these metrics, we gain insights into the model's predictive capabilities and its effectiveness in real-world scenarios. This testing phase ensures that the model meets predefined performance criteria and helps identify areas for potential improvement.

###### 

###### 6.2.2 GENERALIZATION TESTING

Generalization testing in our TensorFlow-based fuel efficiency prediction project assesses how well the model performs on new, unseen data beyond the training set. It ensures that the model can generalize its predictions effectively to diverse instances, reflecting its robustness and reliability in real-world applications. By evaluating performance metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE) on the test dataset, we validate the model's ability to make accurate predictions in different scenarios, essential for deploying it confidently in practical settings.

###### 6.2.3 VALIDATION TESTING

Validation testing in our TensorFlow-based fuel efficiency prediction project verifies the assumptions and decisions made during model development. It focuses on validating preprocessing steps, feature engineering choices, and model configurations to ensure they align with expected outcomes. By assessing the model's performance on validation data using metrics like Mean Absolute Error (MAE) and Mean Squared Error (MSE), we confirm the reliability and effectiveness of our approach. This testing phase ensures that the model is robust and ready for deployment, providing confidence in its predictive capabilities for fuel efficiency estimation.

##### 6.3 TEST CASES

###### 6.3.1 CLASSIFICATION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **Test Case** | **Excepted Result** | Result | **Remarks(IF Fails)** |
| 1. | Test with typical inputs | Predicted class | Passed | - |
| 2. | Test with edge inputs | Predicted class | Passed | -. |
| 3. | Test with outlier inputs | Predicted class | Passed | - |
| 4. | Test with new data | Predicted class | Passed | - |
| 5. | Test with noisy data | Predicted class | Passed | - |

**7.CONCLUSION**

#### 7. CONCLUSION & FUTURE SCOPE

##### PROJECT CONCLUSION

This project successfully demonstrated the use of TensorFlow for predicting fuel efficiency. By processing the Auto MPG dataset, building a neural network model, and performing various testing phases, we achieved reliable predictions. The model's performance was evaluated using metrics like MAE and MSE, and visualizations helped validate its effectiveness. Despite some challenges with edge and noisy data, the model showed strong generalization capabilities. This project underscores the potential of deep learning in predictive analytics, offering a robust approach for estimating fuel efficiency based on vehicle attributes.

##### 7.2 FUTURE SCOPE

Future work can enhance this model by incorporating more advanced architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), to capture complex patterns in the data. Additionally, expanding the dataset with more recent and diverse data could improve model robustness. Techniques like hyperparameter tuning and ensemble learning could further optimize performance. Exploring real-time fuel efficiency prediction systems integrated with IoT devices in vehicles offers a promising application, providing drivers with immediate feedback and contributing to more efficient and eco-friendly driving practices.

### 

### 

### 8.BIBLIOGRAPHY

##### BIBLIOGRAPHY

##### REFERENCES

TensorFlow Documentation

TensorFlow. "TensorFlow Documentation." https://www.tensorflow.org/. Accessed June 2024.

Auto MPG Dataset

Dua, Dheeru, and Efi Karra Taniskidou. "UCI Machine Learning Repository: Auto MPG Data Set." http://archive.ics.uci.edu/ml/datasets/auto+mpg. Accessed June 2024.

Pandas Documentation

Pandas Development Team. "Pandas Documentation." https://pandas.pydata.org/docs/. Accessed June 2024.

Scikit-Learn Documentation

Pedregosa, F., et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, vol. 12, 2011, pp. 2825-2830. https://scikit-learn.org/stable/. Accessed June 2024.

Keras Documentation

Keras Team. "Keras Documentation." https://keras.io/. Accessed June 2024.

Batch Normalization

Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." Proceedings of the 32nd International Conference on Machine Learning (ICML-15), 2015. http://proceedings.mlr.press/v37/ioffe15.html. Accessed June 2024.

Adam Optimizer

Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." arXiv preprint arXiv:1412.6980, 2014. https://arxiv.org/abs/1412.6980. Accessed June 2024.