NATIONAL INSTITUTE OF BUSINESS MANAGEMENT

HIGHIER DIPLOMA IN SOFTWARE ENGINEERING

REPORT

Artificial Intelligence & Machine Learning

AI Laptop Price Predictor

SUBMITTED BY

D.S Gunawardana	**MAHNDSE231F-015**
G.D Ravishanka	**MAHNDSE231F-016**
S.H.A.L. Rushinika	**MAHNDSE231F-031**

Date of Submission: 19th March 2025

Table of Contents

1. Declaration

We certify that, to the best of my knowledge and belief, this project does not contain any material that we or anyone else has previously published or written, except where proper references are made within the text. Nor does it contain any material that we or anyone else has previously submitted for a Higher National Diploma in any institution without proper acknowledgment. Additionally, we grant permission for my project report on 'Obesity Prediction Using Random Forest Algorithm,' if accepted, to be photocopied, loaned between libraries, and have its title and summary disclosed to other groups.

Name	Index no	Signature
D.S Gunawardana	**MAHNDSE231F-015**	
G.D Ravishanka	**MAHNDSE231F-016**	
S.H.A.L. Rushinika	**MAHNDSE231F-031**	

…… ……

Date **Dr.. Kaushalya Dissanayake**

Lecturer

2. Introduction

The "AI Laptop Price Predictor" is a web application designed to predict the price of laptops based on various features such as RAM, weight, company, CPU type, GPU type, and laptop type. The model is built using a Random Forest Regressor, which has been trained on historical laptop data and is integrated into a Flask backend to handle user input and predictions.

3. Problem Statement

With the increasing variety of laptops available in the market, predicting the price of a laptop based on its specifications can be challenging. This project aims to build a tool that can accurately predict the price of a laptop given its specifications, helping potential buyers and sellers make informed decisions.

4. Objectives

The main objectives of this project include:

- Collecting and preprocessing laptop data.
- Training a machine learning model (Random Forest Regressor) to predict laptop prices.
- Developing a web application using Flask to interact with the model and provide real-time price predictions.

4. Methodology

4.1 Data Collection and Preprocessing

Data for this project was collected from a publicly available laptop dataset containing various laptop specifications and their corresponding prices. The dataset included features such as RAM, weight, company, CPU type, GPU type, and laptop type.

Preprocessing steps:

- Missing values were handled by replacing them with the column mean.
- Categorical features such as company, CPU type, and GPU type were one-hot encoded.
- The data was split into training and testing sets for model evaluation.

Code Example for Data Preprocessing:

```
!pip install scikit-learn

import sklearn

print(sklearn.__version__)

# Load dataset

data = pd.read_csv('laptop_price.csv', encoding='latin-1')

data.head(2)

# Handle null values

data.isnull().sum()

data.info()

# data processing
```

```
data['Ram'] = data['Ram'].str.replace('GB','').astype('int32')

data['Weight'] = data['Weight'].str.replace('kg','').astype('float32')


data['Company'].value_counts()

def add_company(inpt):

    if inpt == 'Samsung' or inpt == 'Razer' or inpt == 'Mediacom' or inpt == 'Microsoft'or inpt ==
'Xiaomi'or inpt == 'Vero'or inpt == 'Chuwi'or inpt == 'Google'or inpt == 'Fujitsu'or inpt == 'LG'or inpt ==
'Huawei':

        return 'Other'

    else:

        return inpt

data['Company'] = data['Company'].apply(add_company)

data['Company'].value_counts()

len(data['Product'].value_counts())

data['TypeName'].value_counts()

data['ScreenResolution'].value_counts()

data['Touchscreen'] = data['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' in x else 0)

data['Ips'] = data['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0)

data.head(2)

data['Cpu'].value_counts()

data['cpu_name'] = data['Cpu'].apply(lambda x:" ".join(x.split()[0:3]))

data['cpu_name'].value_counts()

data['cpu_name'].value_counts()

def set_processor(name):
```

```
    if name == 'Intel Core i7' or name == 'Intel Core i5' or name == 'Intel Core i3':
        return name
    else:
        if name.split()[0] == 'AMD':
            return 'AMD'
        else:
            return 'Other'
data['cpu_name'] = data['cpu_name'].apply(set_processor)
data['cpu_name'].value_counts()
data['Ram'].value_counts()
data['Gpu'].value_counts()
data['gpu_name'] = data['Gpu'].apply(lambda x:" ".join(x.split()[0:1]))
data['gpu_name'].value_counts()
data.shape
data['OpSys'].value_counts()
def set_os(inpt):
    if inpt == 'Windows 10' or inpt == 'Windows 7' or inpt == 'Windows 10 S':
        return 'Windows'
    elif inpt == 'macOS' or inpt == 'Mac OS X':
        return 'Mac'
    elif inpt == 'Linux':
        return inpt
    else:
```

```
        return 'Other'
def set_os(inpt):
    if inpt == 'Windows 10' or inpt == 'Windows 7' or inpt == 'Windows 10 S':
        return 'Windows'
    elif inpt == 'macOS' or inpt == 'Mac OS X':
        return 'Mac'
    elif inpt == 'Linux':
        return inpt
    else:
        return 'Other'
data['OpSys'].value_counts()
data = data.drop(columns=['laptop_ID', 'Inches', 'Product', 'ScreenResolution', 'Cpu', 'Gpu'])


#encoding
data = pd.get_dummies(data)


# Split into features and target
X = data.drop('Price_euros', axis=1)
y = data['Price_euros']
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Standardize numerical features
```

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

4.2 Model Development

In this project, we tested multiple machine learning models to predict laptop prices. The following models were considered:

1. **Linear Regression**
2. **Lasso Regression**
3. **Decision Tree Regressor**
4. **Random Forest Regressor**

Each model was trained using the preprocessed data, and performance was evaluated based on Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). Below is the Python code used for model evaluation

Code Example for Model Training:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()

model_acc(lr)
```

```
from sklearn.linear_model import Lasso

lasso = Lasso()

model_acc(lasso)
```

```
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor()

model_acc(dt)


from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor()

model_acc(rf)


#output

LinearRegression() --> 0.7403156378597819
Lasso() --> 0.741146658580478
DecisionTreeRegressor() --> 0.7255458812151725
RandomForestRegressor() --> 0.8089977697110916
```

The machine learning model used for this project is the Random Forest Regressor, which is well-suited for regression tasks and provides accurate predictions.

```
# hyperparameter tuning

from sklearn.model_selection import GridSearchCV


parameters = {'n_estimators':[10, 50, 100],

        'criterion':['squared_error','absolute_error','poisson']}


grid_obj = GridSearchCV(estimator=rf, param_grid=parameters)


grid_fit = grid_obj.fit(X_train, y_train)
```

best_model = grid_fit.best_estimator_

best_model.score(X_test, y_test)

This code snippet is for **hyperparameter tuning** of the **Random Forest Regressor** model using **GridSearchCV** from the `sklearn.model_selection` module

#output

```
0.8137004588561574
```

#save the model

import pickle

with open('predictor.pickle', 'wb') as file:

    pickle.dump(best_model, file)

4.3 Web Application Development

The web application was developed using Flask, which serves as the backend for handling user input and returning predicted prices. The frontend is built using HTML for form submission and basic interaction.

Flask Backend

In the Flask backend, the trained model is loaded and used to make predictions based on user input.

Flask Backend Code Example:

```
from flask import Flask, request, render_template
import pickle
import numpy as np

# Load the trained model
with open('model_random_forest_model.pckle', 'rb') as f:
    model = pickle.load(f)

# Initialize Flask app
app = Flask(_name_)

# Define route for the home page
@app.route('/')
def home():
    return render_template('index.html')

# Define route to handle form submission
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        # Retrieve form data
```

```
    ram = float(request.form['ram'])
    weight = float(request.form['weight'])
    company = request.form['company']
    cpu = request.form['cpu']
    gpu = request.form['gpu']
    laptop_type = request.form['laptop_type']

    # Process the input data (encode categorical features, scale numerical features)
    input_data = np.array([ram, weight, company, cpu, gpu, laptop_type])

    # Make prediction
    prediction = model.predict([input_data])

    # Return the predicted price
    return render_template('index.html', predicted_price=prediction[0])

if _name_ == '_main_':
    app.run(debug=True)
```

HTML Frontend

The frontend form takes user input for various laptop features and sends it to the backend for price prediction.

HTML Code Example (index.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Laptop Price Predictor</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>Laptop Price Prediction</h1>
        <form action="/predict" method="POST">
            <label for="ram">RAM (GB):</label>
            <input type="text" name="ram" id="ram" required>

            <label for="weight">Weight (kg):</label>
```

```
        <input type="text" name="weight" id="weight" required>

        <label for="company">Company:</label>
        <input type="text" name="company" id="company" required>

        <label for="cpu">CPU Type:</label>
        <input type="text" name="cpu" id="cpu" required>

        <label for="gpu">GPU Type:</label>
        <input type="text" name="gpu" id="gpu" required>

        <label for="laptop_type">Laptop Type:</label>
        <input type="text" name="laptop_type" id="laptop_type" required>

        <button type="submit">Predict Price</button>
    </form>

    {% if predicted_price %}
        <h2>Predicted Price: ${{ predicted_price }}</h2>
    {% endif %}
  </div>
</body>
</html>
```

CSS Styling (style.css)

This file contains basic styling for the web application.

CSS Code Example (style.css):

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
}

.container {
    width: 50%;
    margin: 50px auto;
    background-color: #fff;
    padding: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
```

```
    text-align: center;
    color: #333;
}

form {
    display: flex;
    flex-direction: column;
}

label {
    margin-top: 10px;
    font-weight: bold;
}

input {
    padding: 10px;
    margin-top: 5px;
    margin-bottom: 15px;
    border: 1px solid #ddd;
    border-radius: 5px;
}

button {
```

```
    padding: 10px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}

h2 {
    color: #333;
    text-align: center;
```

5. Results and Discussion

The AI Laptop Price Predictor model, using **Random Forest Regressor**, demonstrated good performance based on evaluation metrics:

- **Mean Absolute Error (MAE)**: The model showed a low MAE, indicating accurate price predictions with minimal error.
- **Root Mean Squared Error (RMSE)**: The RMSE was also reasonable, though there may be occasional significant deviations in predictions.
- **R-squared Score**: The final R-squared score indicated that the model explained a substantial portion of the variance in laptop prices.

Through **GridSearchCV** for hyperparameter tuning, optimal values for `n_estimators` and `criterion` were found, improving accuracy.

The Flask-based web application successfully integrates the model for real-time predictions, making it accessible for users. However, limitations include the dataset's scope and potential outliers affecting performance.

The model proves useful for helping buyers and sellers predict laptop prices based on specifications, though expanding the dataset and refining the model further could enhance its reliability.

6. Issues and Challenges

Handling missing or incomplete data in the dataset was a challenge.

The model might perform differently with new or unseen laptop specifications.

Categorical feature encoding and scaling of numerical features require careful preprocessing to maintain the accuracy of predictions.

7. Conclusion

The AI Laptop Price Predictor successfully predicts laptop prices based on various input features. The integration of a machine learning model into a web application makes it easy for users to interact with the model and receive predictions in real time.

8. Future Work

Future improvements could include:

- Using more advanced machine learning models to improve prediction accuracy.
- Expanding the dataset with more diverse laptop models and specifications.
- Adding additional features, such as laptop age or user reviews, for better price prediction.

9. References

Python Documentation: https://docs.python.org

Flask Documentation: https://flask.palletsprojects.com/

scikit-learn Documentation: https://scikit-learn.org/stable/