

Debre Birhan universty

**Collage of computing**

**Department of Software engineering**

**Machine Learning Individual Project**

**Title: Mobile Price Prediction**

**Student Name ID No.**

**Abebaw Addis……………………………….1402806**

**Submitted To:**  **Derbew F. (MSc)**

**Submission Date: 06/02/17 E.C.**

**Project Overview**

This project provides an API for predicting mobile phone prices based on various features, such as battery power, RAM, internal memory, processor speed, and more. The model is trained using a dataset of mobile phone features, and the predictions are made using a **Random Forest Regressor** model. FastAPI serves the model and handles API requests for predictions.

**Technologies Used**

* **Python 3.8+**: The primary programming language for the project.
* **FastAPI**: Web framework for building the RESTful API.
* **scikit-learn**: Machine learning library used for training the model.
* **joblib**: Used to save and load the trained model.
* **pandas**: Data handling and manipulation.
* **seaborn & matplotlib**: For data visualization.
* **uvicorn**: ASGI server for running the FastAPI application.

**API Endpoints**

The API exposes the following endpoints:

**POST /predict**

This endpoint predicts the mobile phone price based on the input features.

**Request Body (JSON)**: The following fields should be provided in the request body:

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| battery\_power | integer | Battery power in mAh (integer). |
| ram | integer | RAM in GB (integer). |
| int\_memory | integer | Internal memory in GB (integer). |
| mobile\_wt | float | Mobile weight in grams (float). |
| processor\_speed | float | Processor speed in GHz (float). |
| fc | integer | Front camera in MP (integer). |
| pc | integer | Primary camera in MP (integer). |
| n\_cores | integer | Number of cores (integer). |
| talk\_time | float | Talk time in hours (float). |
| brand | string | Mobile brand (string). |
| os | string | Operating system (string). |
| touch\_screen | integer | 1 if touch screen supported, 0 otherwise (integer). |
| wifi | integer | 1 if Wi-Fi supported, 0 otherwise (integer). |
| three\_g | integer | 1 if 3G supported, 0 otherwise (integer). |
| four\_g | integer | 1 if 4G supported, 0 otherwise (integer). |
| dual\_sim | integer | 1 if Dual SIM supported, 0 otherwise (integer). |

**Response**: If the prediction is successful, the response will be:

json{

"predicted\_price": 350

}

If there is an error (e.g., missing data or model error), the response will contain an error message:

json{

"error": "Error message"

}

**GET /**

This endpoint serves the static HTML file for the frontend. It can be accessed to display a user interface for input and prediction.

**Response**:  
The response is the index.html file from the template directory.

**Model Overview**

The mobile price prediction model is built using a **Random Forest Regressor** algorithm. Random Forest is an ensemble machine learning method that works by creating multiple decision trees during training and predicting the price by averaging the outputs of those trees.

**Data Preprocessing:**

Before training the model:

1. **Feature Scaling**: Features are standardized using **StandardScaler** to improve the performance of the model.
2. **Handling Missing Values**: The dataset has been cleaned by filling missing values with the mode (most frequent value) of the respective column.
3. **One-Hot Encoding**: Categorical variables such as brand and os are one-hot encoded to make them compatible with the model.

**How the Model Works**

1. **Input Features**: The model takes several input features such as:
   * battery\_power, ram, int\_memory, mobile\_wt, processor\_speed
   * brand, os, touch\_screen, wifi, three\_g, four\_g, dual\_sim, etc.
2. **Preprocessing**:
   * The numerical features are standardized using StandardScaler.
   * The categorical features (brand, os) are encoded using **OneHotEncoder**.
3. **Model**:
   * The **RandomForestRegressor** model is trained using the preprocessed features.
   * The price is the target variable.
   * The model is trained and then saved into a file model.joblib.
4. **Prediction**:
   * When a prediction is requested via the API, the input data is preprocessed (scaling and encoding).
   * The preprocessed data is passed through the trained model to predict the price.
   * The prediction is rescaled back to the original price scale using price\_scaler.

**Model Evaluation:** The model is evaluated using the following metrics:

* **Mean Absolute Error (MAE)**: Measures the average of the absolute errors between predicted and actual values.
* **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual values.
* **Root Mean Squared Error (RMSE)**: The square root of the mean squared error, which provides a more interpretable error metric.
* **R-squared (R²)**: Indicates the proportion of the variance in the target variable that is predictable from the input features.

**Frontend Interface**

The project includes a simple static HTML interface to allow users to input the mobile phone features and get the predicted price. This is located in the template directory and can be accessed by navigating to http://127.0.0.1:8000/template/index.html.

The frontend provides a form where users can input features like battery power, RAM, processor speed, etc. Upon submission, it makes a request to the /predict endpoint and displays the predicted price.

**Model Deployment**

Once you’ve verified the application locally, you can deploy it to a cloud platform for production use. Common platforms for deploying FastAPI applications include:

* **Heroku**: Set up with a Procfile and deploy via GitHub.
* **Render**: Create a web service and connect your repository.
* **AWS EC2** or **AWS Lambda**: For custom hosting.

For example, on **Render**:

1. Create a new "Web Service" and link it to your GitHub repository.
2. Set the build command to pip install -r requirements.txt and the start command to uvicorn fastapi\_app:app --host 0.0.0.0 --port 80.
3. Deploy the service.

**Deployed Model**

The model is deployed on **Render** and can be accessed at the following URL:

* **API URL**: <https://machine-learning1-719x.onrender.com/>
* **Github URL:** https://github.com/ /Abebaw-Addis/machine\_learning1.git

## Notes

* This API is designed to predict the price of mobile phones based on various technical features. It uses machine learning (Random Forest Regressor) for price prediction.
* The API accepts data in JSON format and returns the predicted price in JSON format.
* The model has been trained on a dataset containing features such as battery power, RAM, processor speed, camera megapixels, and more.
* The deployment on **Render** ensures that the model is easily accessible via the internet for integration with other applications.

**Future Improvements**

1. **Model Optimization**: Experiment with other models such as **XGBoost**, **LightGBM**, or **Support Vector Machines** to improve prediction accuracy.
2. **Real-Time Predictions**: Implement a caching mechanism to handle real-time predictions efficiently.
3. **Frontend Enhancements**: Develop a more advanced frontend using frameworks like **React** or **Vue.js** for a more interactive user experience.
4. **Advanced Input Validation**: Add more robust validation for inputs to ensure they follow correct formats and ranges.
5. **Cross-validation**: Implement cross-validation during training to prevent overfitting and improve model generalization.

**Contact Information**

For any questions or suggestions, feel free to contact me at:

* **Email**: abadis1221@gmail.com
* **GitHub**: https://github.com/Abebaw-Addis