**Car Dekho - Used Car Price Prediction using Machine Learning**

**Introduction:**

**Car Dekho - Used Car Price Prediction** is a machine learning-based project aimed at predicting the prices of used cars. The platform is designed for evaluating used car prices, empowering buyers, sellers, and dealers to make informed decisions. This project utilizes machine learning for price prediction and Streamlit for interactive web application development. By analysing car-related data, the application offers instant price estimates, enhancing transparency and efficiency in the used car market.

**Project Scope:**

Based on the historical data on used car prices from CarDekho, including various features such as make, model, year, fuel type, transmission type, and other relevant attributes from different cities. The task is to develop a machine learning model that can accurately predict the prices of used cars based on these features. The model should be integrated into a Streamlit-based web application to allow users to input car details and receive an estimated price instantly.

**Package Used:**

* **Pandas:**
* Pandas is a powerful and open-source Python library. The Pandas library is used for data manipulation and analysis.
* Pandas consist of data structures and functions to perform efficient operations on data.
* To know more about **Pandas**. Click here [https://pandas.pydata.org/docs/](https://pandas.pydata.org/docs/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20)
* **Streamlit:**
* Streamlit turns data scripts into shareable web apps in minutes. All in pure Python. No front‑end experience required.
* To know more about **Streamlit**. Click here <https://docs.streamlit.io/>
* **Streamlit-option-menu:**
* Streamlit-option-menu is a simple Streamlit component that allows users to select a single item from a list of options in a menu.
* To know more about **Streamlit-Option-Menu**. Click here [https://discuss.streamlit.io/t/streamlit-option-menu-is-a-simple-streamlit-component-that-allows-users-to-select-a-single-item-from-a-list-of-options-in-a-menu/20514](https://discuss.streamlit.io/t/streamlit-option-%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20menu-is-a-simple-streamlit-component-that-allows-users-to-select-a-single-item-from-a-list-of-options-in-a-menu/20514)
* **Matplotlib:**
* Matplotlib is a powerful data visualization library for Python. It provides a wide range of tools for creating static, animated, and interactive plots, making it a popular choice for visualizing data in fields like data science, machine learning, and scientific research.
* To know more about **Matplotlib**. Click here <https://matplotlib.org/>
* **Seaborn:**
* Seaborn is a Python data visualization library based on [matplotlib](https://matplotlib.org/). It provides a high-level interface for drawing attractive and informative statistical graphics.
* To know more about **Seaborn**. Click here <https://seaborn.pydata.org/>
* **Scikit-learn:**
  + Simple and efficient tools for predictive data analysis
  + Accessible to everybody, and reusable in various contexts
  + Built on NumPy, SciPy, and matplotlib
  + To know more about **Scikit-learn**. Click here <https://scikit-learn.org/stable/>
* **Numpy:**
* NumPy is the fundamental package for scientific computing in Python.
* It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
* To know more about **Numpy**. Click here <https://numpy.org/doc/stable/>
* **SciPy:**
* SciPy is a scientific computation library that uses [NumPy](https://www.w3schools.com/python/numpy/default.asp) underneath.
* SciPy stands for Scientific Python.
* It provides more utility functions for optimization, stats and signal processing.
* Like NumPy, SciPy is open source so we can use it freely.
* To know more about **SciPy**. Click here <https://docs.scipy.org/doc/scipy/>
* **XGBoost:**
* XGBoost, or Extreme Gradient Boosting, is an open-source machine learning library that uses gradient boosted decision trees (GBDTs) to train models for regression, classification, and ranking tasks
* To know more about **XGBoost.** Click here<https://xgboost.readthedocs.io/en/stable/install.html>
* **Joblib:**
* Joblib is a set of tools to provide lightweight pipelining in Python.
* To know more about **SciPy**. Click here <https://joblib.readthedocs.io/en/stable/>

**Install Packages:**

* **Numpy** – pip install numpy
* **Pandas** – pip install pandas
* **Streamlit** – pip install streamlit
* **Streamlit-Option-Menu** – pip install streamlit-option-menu
* **Joblib** – pip install joblib
* **SciPy** – pip install scipy
* **Seaborn** – pip install seaborn
* **Scikit** **learn** - pip install scikit-learn
* **Matplotlib** – pip install matplotlib
* **XGBoost** – pip install xgboost

**Code Flow Plan:**

This project contains Seven files

1. process\_car\_data\_1.py
2. data\_cleaning\_2.ipynb
3. domain\_data\_model\_3.ipynb
4. UI\_streamlit.py
5. styles.css
6. encoded\_mappings.pkl
7. carprice\_prediction\_ml\_model.pkl

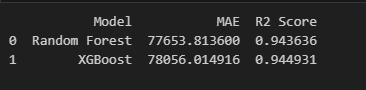
* **process\_car\_data\_1.py**
* The script preprocesses datasets from six cities and merges them into a single file (Merged\_Cities\_data.csv).
* It imports necessary libraries: pandas for data manipulation, numpy for handling missing values, and ast for parsing strings into dictionaries.
* City-specific datasets are loaded from Excel files using pandas.read\_excel.
* Nested data in columns is flattened using dedicated functions for car details, features, overview, and specifications.
* Each flattening function (convert\_strdict\_to\_pydict, flatten\_new\_car\_overview, flatten\_new\_car\_feature, flatten\_new\_car\_specs) extracts key information into a structured format.
* The process\_city\_dataset function processes data for each city, applying flattening functions and concatenating results with city-specific links.
* Individual processed datasets are saved as CSV files for each city in the Flattened\_dataset directory.
* The combine\_files function merges all city-specific CSV files into a single structured file named Merged\_Cities\_data.csv.
* The script includes error handling in all major functions to log and handle exceptions gracefully.
* Outputs include both individual city-specific CSV files and the final merged dataset for further analysis or modeling.
* **data\_cleaning\_2.ipynb**
* The notebook imports libraries like pandas, matplotlib, seaborn, and scipy.stats for data manipulation, visualization, and statistical analysis.
* Reads the Merged\_Cities\_data.csv file and displays dataset info, column names, and initial size for understanding the structure.
* Drops unnecessary or duplicate columns after checking for redundancies between columns like ownerNo and owner, and standardizes column names.
* Formats columns by removing extra characters (e.g., cc, @rpm, kms) and ensures numeric data is converted to appropriate data types.
* Handles missing values by dropping columns/rows with more than 50% nulls and imputing others using statistical methods like mean or mode.
* Identifies and removes duplicate rows while ensuring the integrity of unique entries.
* Detects and corrects misspelled categorical values using mappings (e.g., oem, Insurance Validity) to maintain consistency.
* Performs exploratory data analysis (EDA) using visualizations like histograms, count plots, and correlation analysis with the target variable (price).
* Detects outliers using methods like IQR and Z-scores, then handles them using Winsorization to cap extreme values.
* Selects key features based on correlation and domain knowledge, exporting the cleaned and formatted data as domain\_related\_data.csv for model deployment.
* **domain\_data\_model\_3.ipynb**
* The dataset domain\_related\_data.csv is loaded, and its categorical and continuous columns are identified.
* Categorical columns are encoded using LabelEncoder, and mappings are stored for reference in model deployment.
* The target variable price is separated from features, and the dataset is split into training and testing sets using an 80-20 ratio.
* Multiple machine learning models (e.g., Linear Regression, Decision Tree, Random Forest, XGBoost) are initialized and evaluated.
* Random Forest and XGBoost are identified as the best-performing models based on Mean Absolute Error (MAE) and R² scores.
* Hyperparameter tuning is performed using RandomizedSearchCV for both models to optimize parameters such as n\_estimators, max\_depth, and learning\_rate.
* Feature importance is plotted for Random Forest and XGBoost to analyze which features significantly impact predictions.
* The best XGBoost model achieves high R² and low MAE, with parameters including n\_estimators=200, max\_depth=7, and learning\_rate=0.05.
* The trained XGBoost model is saved as carprice\_prediction\_ml\_model.pkl for deployment.
* The notebook concludes by preparing the model for integration into the Streamlit application using the best parameters.
* Save the model (carprice\_prediction\_ml\_model.pkl) and encoded mapping (encoded\_mappings.pkl) using joblib
* **UI\_streamlit.py**
* Loads the dataset (domain\_related\_data.csv), encoded mappings, and the trained machine learning model (carprice\_prediction\_ml\_model.pkl) using pandas and joblib.
* Configures a Streamlit application with a title, icon, and wide layout, providing an interactive interface for car price prediction.
* Implements a sidebar menu using streamlit\_option\_menu with options for **Home** and **Car Price Predict**, styled with custom CSS.
* On the **Home** page, introduces the application and explains its features, benefits, and how it works using Markdown and styled headings.
* On the **Car Price Predict** page, allows users to input car details via dropdowns and numeric input fields for attributes like brand, model, transmission type, engine displacement, mileage, and gear count.
* Validates user inputs to ensure all required fields are selected before proceeding with prediction.
* Encodes categorical inputs using pre-loaded mappings and prepares them for model prediction in a structured DataFrame format.
* Predicts the car price using the trained machine learning model and displays the result with formatted styling, including the predicted price in rupees.
* Includes a **Reset** button to clear all input fields and reload the interface for a fresh start.
* Enhances user experience with custom CSS for buttons, dropdowns, and input fields, ensuring a visually appealing and user-friendly interface.

**How to run the code:**

* Initially run the “process\_car\_data\_1.py” using command “python process\_car\_data\_1.py” in terminal.
* Next run “data\_cleaning\_2.ipynb” by selecting “Run All”
* Next run “domain\_data\_model\_3.ipynb” by selecting “Run All”
* Next run “UI\_streamlit.py” by using command “streamlit run UI\_streamlit.py”

**Model Score:**

Score when training the model using default params

****

Score after training model with best params

****

**Steamlit UI:**

