SAMA Tasks Report

Task:- 1

1. Dataset Overview

The dataset used for this analysis is "car_data.csv". The initial steps involved loading the dataset and inspecting its structure:

- **df.head()**: Displays the first five rows of the dataset.
- **df.info()**: Provides a concise summary of the dataset, including the data types of each column and the non-null counts.
- **df.describe()**: Shows basic statistical details like mean, standard deviation, min, and max values for numerical columns.

2. Data Cleaning

To ensure the dataset was clean and ready for model training, the following steps were taken:

- **Missing Values**: Checked for missing values using **df.isnull().sum()**.
- **Duplicated Rows**: Identified duplicated rows using **df.duplicated().sum()**.
- **Removal of Missing and Duplicate Entries**: Used **df.dropna(inplace=True)** and **df.drop_duplicates(inplace=True)** to remove rows with missing values and duplicates, respectively.
- **Reset Index**: Reset the index of the DataFrame to maintain order using **df.reset_index(drop=True, inplace=True)**.

3. Handling Special Cases in Data

There were rows where the 'Price' column had the value "Call for price". These rows were counted and then removed to avoid errors in numerical processing:

- Counted the special cases using **(df['Price'] == 'Call for price').sum()**.
- Removed these rows using **df = df[df['Price'] != 'Call for price']**.
- Converted the 'Price' column to numeric using **df['Price'] = pd.to_numeric(df['Price'])**.

4. Encoding Categorical Variables

To convert categorical variables into numerical format, one-hot encoding was applied:

- Used **pd.get_dummies** to create dummy variables for categorical columns like 'Make', 'Model', 'Version', 'Assembly', 'Registered City', and 'Transmission'.

5. Feature Scaling

Scaled numerical features to ensure uniformity across features using **StandardScaler**:

- Features scaled: 'Make_Year', 'CC', 'Mileage'.
- Applied scaling with **scaler.fit_transform(df[features_to_scale])**.

6. Feature Engineering

Created a new feature 'Car_Age' from 'Make_Year':

- Calculated as **2024 - df['Make_Year']**.
- Dropped the original 'Make_Year' column using **df.drop('Make_Year', axis=1, inplace=True)**.

7. Data Splitting

Split the data into training and testing sets:

- Used **train_test_split** with a test size of 20% and a random state of 42 to ensure reproducibility.

8. Model Training and Evaluation

Trained and evaluated multiple regression models:

- Models used: Linear Regression, Decision Tree Regressor, Random Forest Regressor, Gradient Boosting Regressor.
- Evaluated models using RMSE and R^2 score.

Conclusion

For the first task, I reviewed the dataset which included identifying and handling missing and duplicate values. After cleaning the dataset, one-hot encoding was

applied to convert categorical variables into numerical formats, followed by feature scaling to ensure uniformity across features. Feature engineering was also performed to create a new feature 'Car_Age'. After splitting the dataset, various regression models were trained and evaluated. The model with the best R^2 score was saved for future use. Hypertuning the best model could be considered, but it might lead to overfitting.

TASK: 2

1. Dataset Overview

The dataset used for this analysis is "movies_dataset.csv". The initial steps involved loading the dataset and inspecting its structure:

- Loaded the dataset using **pd.read_csv("movies_dataset.csv")**.
- Displayed the dataset using **data**.

2. Data Inspection and Cleaning

To ensure the dataset was clean and ready for model training, the following steps were taken:

- **Descriptive Statistics**: Generated summary statistics using **data.describe()**.
- **Missing Values**: Checked for missing values using **data.isnull().sum()**.
- **Duplicated Rows**: Identified duplicated rows using **data.duplicated().sum()**.
- **Handling Movie Titles**: Extracted the 'title' column into a separate variable **movie_t** and removed it from the main dataset using **data.drop(columns=["title"])**.

3. Data Preprocessing

- **Scaling Features**: The features were scaled to ensure uniformity:
- **Data Splitting:** Split the dataset into training and testing sets using train_test_split with a test size of 20% and a random state of 42.

4. Cosine Similarity for Recommendation

- **Cosine Similarity Calculation**: Computed the cosine similarity matrix between the test and train sets:
- **Recommendation Function**: Created a function to recommend a movie based on the cosine similarity:

5. Nearest Neighbors for Recommendation

- **Nearest Neighbors Model**: Implemented the Nearest Neighbors model with cosine distance
- **Recommendation Function**: Modified the recommendation function to use the Nearest Neighbors model:

Conclusion

During the analysis, multiple recommendation models were tested, but the two most reliable models were based on cosine similarity and Nearest Neighbors. The dataset had some challenges, including missing and duplicated values, which were addressed during preprocessing. After cleaning the data and scaling the features, the recommendation models were implemented. The cosine similarity model and Nearest Neighbors model both provided satisfactory recommendations based on the given input movie.