|  |  |
| --- | --- |
| **Shri Vishnu Engineering College for Women (Autonomous)** | |
| **Department of CSE** | |
| **Course Details** | |
| **Regulation** | **R22** |
| **Year / Semester** | **III B.Tech – II Sem** |
| **Course** | **Data Science with R Programming (Theory & Lab)** |
| **Course Code** | **UGCS6T0822** |
| **Course Type** | **Job Oriented Elective ( JOE )** |
| **Faculty** | **Y.Ramu – Department of CSE** |

|  |  |
| --- | --- |
| **Student Details** | |
| **Section** | **CSE-CS** |
| **Registered Number** | **22B01A4612** |
| **Student Name** | **Ch Sai Satvika** |
| **Mobile Number** | **8121355340** |
| **Email ID (Gmail)** | **satvika1609@gmail.com** |

|  |  |
| --- | --- |
| **Case Study Details** | |
| **Domain** | **E-commerce** |
| **Title of the Case Study** | **Perfume Price Prediction** |
| **Tools Used** | **Python** |
| **Date of Verification** |  |
|  |  |

|  |
| --- |
| **Name of the Dataset: eBay Men’s Perfume Dataset** |
| **Dataset URL (Active in online):**  https://www.kaggle.com/datasets/kanchana1990/perfume-e-commerce-dataset-2024 |
| **Dataset Description**:  The dataset consists of various perfume products listed on eBay. It contains information such as brand, perfume type, price, availability, sales, and item location. |
| **Features in Dataset: (include all feature names and their descriptions as per the information available at the source of dataset (Kaggle / UCI Data Repository etc)**   |  |  | | --- | --- | | Brand | Name of the perfume brand |  |  |  | | --- | --- | | type | Type of perfume (Eau de Parfum, Eau de Toilette, etc.) |  |  |  | | --- | --- | | price | Selling price of the perfume |  |  |  | | --- | --- | | available | Number of available stock units |  |  |  | | --- | --- | | sold | Number of units sold |  |  |  | | --- | --- | | itemLocation | Location of the seller | |
| **Number of Features in Dataset: 6** |
| **Number of Samples (records) in Dataset: 1000** |
| **Is the dataset is having null values: Yes** |
| **Is the dataset is having missing values: Yes** |
| **Is the dataset is in encoded format of PCA values: No** |
| **Is it essential to pre-process the dataset for the case study: Yes**  **If Yes, how you want to preprocess? Give details:**  1.Handle Missing Values:   * Fill categorical (brand, type, itemLocation) with mode. * Fill numerical (available, sold) with median.   2.Drop Unnecessary Columns:   * Remove priceWithCurrency, availableText, lastUpdated, title.   3.Encode Categorical Variables:   * Convert brand, type, itemLocation using Label Encoding.   4.Feature Selection:   * Target: price * Features: brand, type, available, sold, itemLocation   5.Split Dataset:   * Train-test ratios: 80:20, 75:25, 70:30 |
| **List out the possible opportunities for analysis on this dataset based on the available features**   * Predicting the selling price of a perfume based on brand, type, availability, and location. * Forecasting demand based on historical sales and availability trends. * Understanding which features influence perfume pricing the most. |

|  |
| --- |
| **Title of the Case Study: Perfume Price Prediction Using Machine Learning** |
| **List of Objectives:**   1. Perform **data preprocessing** to handle missing values and categorical encoding. 2. Apply **various regression models** to predict the price of perfumes. 3. Compare different models using evaluation metrics (**R² Score, MAE, RMSE**). 4. Identify **key factors influencing perfume pricing** using feature importance. 5. Visualize predictions and analyze residual errors. |
| **Approach: What features are going to be considered, processed, or feature-engineered to derive a specific outcome after applying one or more models?**   * Considered brand, type, availability, sold quantity, and item location as input features. * Processed missing values and encoded categorical features. * Applied 5 regression models to predict price. * Evaluated models using different train-test splits (80:20, 75:25, 70:30) |

|  |
| --- |
| **Methodology: List out the overall implementation plan of your case study in step-by-step approach. (Data Preprocessing, Feature selection, Feature engineering, model selection, model building, model training approach, model testing, evaluation of metrics etc)** |
| **1. Data Preprocessing**   * Load dataset and inspect for missing values. * Handle missing values (fill missing categorical values with mode, numerical values with median). * Encode categorical variables using Label Encoding. * Drop unnecessary columns.   **2. Feature Selection & Engineering**   * Define target (price) and features (brand, type, available, sold, itemLocation).   **3. Train-Test Splitting**   * Split the dataset into 80:20, 75:25, 70:30 ratios.   **4. Model Selection**   * Apply Linear Regression, Decision Tree, Random Forest, Support Vector Regression, and Gradient Boosting.   **5. Model Training & Evaluation**   * Train each model on different train-test splits. * Compute R² Score, MAE, and RMSE.   **6. Visualization**   * Feature importance plot (Random Forest). * Actual vs. Predicted scatter plot. * Residuals distribution |

**Case-Study Implementation**

|  |  |
| --- | --- |
| Task | Loading Dataset |
| Step-1 | Loading Dataset with name, display its descriptive information |
| Description | The dataset is loaded and its basic information is displayed. |
| Code | import pandas as pd  df = pd.read\_csv("ebay\_mens\_perfume.csv")  df.info()  df.head() |
| Result | # Column Non-Null Count Dtype  --- ------ -------------- -----  0 brand 999 non-null object  1 title 1000 non-null object  2 type 997 non-null object  3 price 1000 non-null float64  4 priceWith 1000 non-null object  Currency  5 available 889 non-null float64  6 availableText 997 non-null object  7 sold 994 non-null float64  8 lastUpdated 947 non-null object  9 itemLocation 1000 non-null object |
| Description about results in detailed way | * The dataset contains **1000 records** and **10 columns**. * Some features (brand, type, available, sold, lastUpdated) have missing values. * The target variable for regression is **price**, which is a numerical column. * Some columns like **title, priceWithCurrency, and lastUpdated** are **not useful for modeling** and will be removed in preprocessing. |

|  |  |
| --- | --- |
| Task | Data Pre-processing |
| Step -2 | Dropping Parameters / Columns |
| Description | Before applying machine learning models, the dataset must be **cleaned and prepared**:  1. **Drop Unnecessary Columns**   * title: Not needed for price prediction. * priceWithCurrency: Redundant as we have price. * availableText: Text format, not useful for modeling. * lastUpdated: Not relevant for prediction.   2. **Handle Missing Values**   * **Categorical Features** (brand, type, itemLocation): Fill with the most frequent value (mode). * **Numerical Features** (available, sold): Fill with the median value.   3. **Encode Categorical Features**   * Convert brand, type, and itemLocation into numerical format using **Label Encoding**. |
| Code | from sklearn.preprocessing import LabelEncoder  from sklearn.impute import SimpleImputer  # Drop unnecessary columns  df\_cleaned = df.drop(columns=["priceWithCurrency", "availableText", "lastUpdated", "title"])  # Handle missing values  imputer = SimpleImputer(strategy="most\_frequent")  df\_cleaned[["brand", "type", "itemLocation"]] = imputer.fit\_transform(df\_cleaned[["brand", "type", "itemLocation"]])  df\_cleaned[["available", "sold"]] = df\_cleaned[["available", "sold"]].fillna(df\_cleaned[["available", "sold"]].median())  # Encode categorical variables  label\_encoders = {}  for col in ["brand", "type", "itemLocation"]:  le = LabelEncoder()  df\_cleaned[col] = le.fit\_transform(df\_cleaned[col])  label\_encoders[col] = le  df\_cleaned.head() |
| Result | brand type price available sold itemLocation  0 68 25 84.99 10.0 116.0 5  1 3 25 109.99 8.0 48.0 13  2 223 29 100.00 10.0 27.0 59  3 96 29 44.99 2.0 159.0 210  4 141 44 16.91 10.0 156.0 25 |
| Description about results in detailed way | * The dataset is **cleaned and ready** for modeling. * Categorical features (brand, type, itemLocation) are now **numerical**. * Missing values have been filled, ensuring no data loss. * Unnecessary columns have been dropped. |

|  |  |
| --- | --- |
| Task | Splitting the Dataset |
| Step | Splitting the Dataset |
| Description | We split the dataset into **train and test sets** to evaluate model performance. The following ratios are used:   1. **80:20** (80% training, 20% testing) 2. **75:25** (75% training, 25% testing) 3. **70:30** (70% training, 30% testing) |
| Code | from sklearn.model\_selection import train\_test\_split  split\_ratios = {"80:20": 0.2, "75:25": 0.25, "70:30": 0.3}  splits = {}  for split\_name, test\_size in split\_ratios.items():  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=test\_size, random\_state=42)  splits[split\_name] = (X\_train, X\_test, y\_train, y\_test)  # Print dataset sizes  for split, (X\_train, X\_test, y\_train, y\_test) in splits.items():  print(f"{split} -> Train: {len(X\_train)}, Test: {len(X\_test)}") |
| Result | 80:20 -> Train: 800, Test: 200  75:25 -> Train: 750, Test: 250  70:30 -> Train: 700, Test: 300 |
| Description about results in detailed way | * The dataset has been successfully split into different ratios. * This ensures the models are trained on sufficient data and tested on unseen data. |

|  |  |
| --- | --- |
| Task | Training Regression Models |
| Step | Training Regression Models |
| Description | We train **5 different regression models** to predict perfume prices:   1. **Linear Regression** 2. **Decision Tree Regression** 3. **Random Forest Regression** 4. **Support Vector Regression (SVR)** 5. **Gradient Boosting Regression**   Each model is trained and evaluated using:   * **R² Score** (Measures how well the model fits the data) * **MAE** (Mean Absolute Error – Measures average error) * **RMSE** (Root Mean Squared Error – Penalizes large errors) |
| Code | from sklearn.ensemble import RandomForestRegressor  # Train Random Forest on 80:20 split  X\_train, X\_test, y\_train, y\_test = splits["80:20"]  model = RandomForestRegressor(n\_estimators=100, random\_state=42)  model.fit(X\_train, y\_train)  # Predictions  y\_pred = model.predict(X\_test)  # Evaluate model  r2 = r2\_score(y\_test, y\_pred)  mae = mean\_absolute\_error(y\_test, y\_pred)  rmse = mean\_squared\_error(y\_test, y\_pred, squared=False)  print(f"R² Score: {r2}, MAE: {mae}, RMSE: {rmse}") |
| Result | R² Score: 0.1289, MAE: 23.15, RMSE: 35.67 |
| Description about results in detailed way | * **Random Forest Regression performed best** with **R² = 0.1289**. * The model is **not highly accurate**, suggesting more feature engineering is needed. * Future improvements could include **hyperparameter tuning** and **additional features.** |