|  |  |
| --- | --- |
| **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-Cyber Security** |
| **Registered Number** | **22B01A4609** |
| **Student Name** | **B. Krishna Raja Sree** |
| **Mobile Number** | **9063643676** |
| **Email ID (Gmail)** | **sreebonam555@gmail.com** |

|  |  |
| --- | --- |
| **Case Study Details** | |
| **Domain** | **Banking** |
| **Title of the Case Study** | **Default of Credit Card Clients** |
| **Tools Used** | **Python** |
| **Date of Verification** |  |
|  |  |

|  |
| --- |
| **Name of the Dataset:** UCI\_Credit\_Card |
| * **Dataset URL (Active in online):** * https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset |
| **Dataset Description**:  This dataset contains **30,000 records** of credit card users, including details about their **credit limit, past payments, bill amounts, and whether they defaulted**.  **Target Variable:** "default.payment.next.month" (0 = No Default, 1 = Default) |
| **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)**   * **ID**: Unique client identifier * **LIMIT\_BAL**: Credit limit amount * **SEX**: Gender (1 = Male, 2 = Female) * **EDUCATION**: Education level (1 = Graduate, 2 = University, etc.) * **MARRIAGE**: Marital status (1 = Married, 2 = Single, etc.) * **AGE**: Age of the client * **PAY\_0 to PAY\_6**: Payment history in the last 6 months (indicates whether the client paid on time or had late payments) * **BILL\_AMT1 to BILL\_AMT6**: Past 6 months' bill amounts * **PAY\_AMT1 to PAY\_AMT6**: Amount paid in the last 6 months * **default.payment.next.month**: **Target Variable** (0 = No Default, 1 = Default) |
| **Number of Features in Dataset:23** |
| **Number of Samples (records) in Dataset:30,000** |
| **Is the dataset is having null values: No** |
| **Is the dataset is having missing values: No** |
| **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:**   * Convert categorical variables (SEX, EDUCATION, MARRIAGE) using pd.get\_dummies() * Standardization: Scale numeric features using StandardScaler() * Handle class imbalance: Use SMOTE (as default cases are imbalanced) * Feature Engineering: Remove unnecessary columns (ID), analyze feature importance |
| **List out the possible opportunities for analysis on this dataset based on the available features**   * Predict which customers are likely to default on their credit card payments * Identify important financial factors affecting defaults * Optimize credit policies by analyzing risk patterns * Compare ML models for the best default prediction system |

|  |
| --- |
| **Title of the Case Study:** Credit Card Default Prediction |
| **List of Objectives:**   * Build a model to predict credit card default * Preprocess the dataset to handle categorical and numeric features * Use Logistic Regression and XGBoost for classification * Evaluate model performance using accuracy, confusion matrix, and classification report |
| **Approach: What features are going to be considered, processed, or feature-engineered to derive a specific outcome after applying one or more models?**   * Consider all financial & demographic features (LIMIT\_BAL, PAY\_X, etc.) * Convert categorical variables (SEX, EDUCATION, MARRIAGE) to numerical * Standardize features using StandardScaler() * Handle imbalanced data using SMOTE |

|  |
| --- |
| **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)** |
| * Step 1: Load the dataset * Step 2: Handle categorical variables (pd.get\_dummies()) * Step 3: Remove unnecessary features (ID) * Step 4: Split data into train & test sets (80-20 split) * Step 5: Apply StandardScaler() for normalization * Step 6: Handle class imbalance using SMOTE * Step 7: Train Logistic Regression and XGBoost models * Step 8: Evaluate using accuracy, confusion matrix, classification report * Step 9: Visualize results (Confusion Matrix, Feature Importance) |

**Case-Study Implementation**

* Provide complete implementing details with descriptions for every step of the case study as per the above methodology.
* This entire implementation must be available in a single code file.
* Results of the case study must justify the objectives.
* Results must be included in EXCEL file with various performance metrics under different training & testing ratio combinations (or) as suggested by the faculty.
* Code & Results must be verified by the faculty before submission of the case-study

|  |  |
| --- | --- |
| Task | Loading Dataset |
| Description | The dataset is loaded and displayed. |
| Code | import pandas as pd  from google.colab import drive  # Mount Google Drive  drive.mount('/content/drive')  # Load dataset  file\_path = "/content/drive/My Drive/credit\_card\_default.csv"  data = pd.read\_csv(file\_path)  # Display first 5 rows  print(data.head()) |
| Result | Dataset loaded successfully. |

|  |  |
| --- | --- |
| Task | Data Pre-processing |
| Description | Handle categorical features and remove unnecessary columns. |
| Code | # Drop the "ID" column  data = data.drop(columns=["ID"])  # Rename target variable for clarity  data = data.rename(columns={"default.payment.next.month": "Default"})  # Convert categorical features to numerical  categorical\_cols = ["SEX", "EDUCATION", "MARRIAGE"]  data = pd.get\_dummies(data, columns=categorical\_cols, drop\_first=True)  # Display updated dataset  print(data.head()) |
| Result | Categorical features converted. |

|  |  |
| --- | --- |
| Task | Train-Test Split & Scaling |
| Description | Splitting data and applying StandardScaler(). |
| Code | from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  # Define features & target  X = data.drop("Default", axis=1)  y = data["Default"]  # Split dataset  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)  # Standardization  scaler = StandardScaler()  X\_train = scaler.fit\_transform(X\_train)  X\_test = scaler.transform(X\_test) |
| Result | Data split & scaled. |

|  |  |
| --- | --- |
| Task | Handle Imbalanced Data Using SMOTE |
| Description | Apply SMOTE to handle class imbalance. |
| Code | from imblearn.over\_sampling import SMOTE  # Apply SMOTE  smote = SMOTE(random\_state=42)  X\_train\_smote, y\_train\_smote = smote.fit\_resample(X\_train, y\_train)  # Print new class distribution  print(y\_train\_smote.value\_counts()) |
| Result | Imbalance handled with **SMOTE**. |

|  |  |
| --- | --- |
| Task | Train Logistic Regression Model |
| Description | Train and evaluate Logistic Regression. |
| Code | from sklearn.linear\_model import LogisticRegression  from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report  # Train model  model = LogisticRegression()  model.fit(X\_train\_smote, y\_train\_smote)  # Predict  y\_pred = model.predict(X\_test)  # Evaluate  print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.2f}")  print(confusion\_matrix(y\_test, y\_pred))  print(classification\_report(y\_test, y\_pred)) |
| Result | Model trained with **~70% accuracy** |

|  |  |
| --- | --- |
| Task | Train XGBoost Model |
| Description | Train XGBoost for better performance. |
| Code | from xgboost import XGBClassifier  # Train XGBoost model  xgb\_model = XGBClassifier(use\_label\_encoder=False, eval\_metric="logloss")  xgb\_model.fit(X\_train\_smote, y\_train\_smote)  # Predict  y\_pred\_xgb = xgb\_model.predict(X\_test)  # Evaluate  print(f"XGBoost Accuracy: {accuracy\_score(y\_test, y\_pred\_xgb):.2f}") |
| Result | XGBoost gives **~80 accuracy**. |

Add more tables as per the number steps in your project

**Have a folder with your register number with following files & names to submit as single .rar fille**

**Folder name: Regdnumber Casestudy**

**Example: 22B1A0575 FAANG Stock Price Prediction**

|  |  |  |
| --- | --- | --- |
| **File Type** | **Filename** | **Example** |
| Dataset | as per the original name when you downloaded | FAANG |
| Code Files | (one or more as applicable)  as per case study name | FAANG\_StockPrice\_Prediction V1 20.02.25  FAANG\_StockPrice\_Prediction V2 25.02.25 |
| Results File with graphs (in Excel Format) | as per case study name | FAANG\_StockPrice\_Prediction\_Results |
| Case Study Report | as per case study name | FAANG\_StockPrice\_Prediction\_Report |
| Other Relevant File |  |  |