|  |  |
| --- | --- |
| **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-B** |
| **Registered Number** | **22B01A0575** |
| **Student Name** | **KANKIPATI SOWMYASRI** |
| **Mobile Number** | **8463983408** |
| **Email ID (Gmail)** | **kankipatisowmyasri@gmial.com** |

|  |  |
| --- | --- |
| **Case Study Details** | |
| **Domain** | **Finance** |
| **Title of the Case Study** | **Predicting whether the customers exited the bank or not** |
| **Tools Used** | **Python** |
| **Date of Verification** | **03/03/2025** |
|  |  |

|  |
| --- |
| **Name of the Dataset: Churn\_Modelling.csv"** |
| **Dataset URL (Active in online):** [**https://www.kaggle.com/datasets/aakash50897/churn-modellingcsv**](https://www.kaggle.com/datasets/aakash50897/churn-modellingcsv)  **Chatgpt conversation link :** [**https://chatgpt.com/share/67d459e3-c3cc-8005-b703-54a89c7081bf**](https://chatgpt.com/share/67d459e3-c3cc-8005-b703-54a89c7081bf) |
| **Dataset Description**:  The dataset contains information about **10,000** bank customers and **14 features**, including demographic, financial, and behavioral attributes. The goal is to predict whether a customer will **churn (exit the bank) or remain** using various features.  **Target Variable**   * **Exited** *(Binary: 0 or 1)*   + 0 → Customer **stayed** with the bank   + 1 → Customer **churned** (left the bank) |
| **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)**   1. Row\_Number 2. Customer\_ID 3. Surname 4. CreditScore 5. Geography 6. Gender 7. Age 8. Tenure 9. Balance 10. NumOfProducts 11. HasCrCard 12. IsActiveMember 13. EstimatedSalary 14. Exited |
| **Number of Features in Dataset: 14** |
| **Number of Samples (records) in Dataset: 10,000** |
| **Is the dataset is having null values: No** |
| **Is the dataset is having missing values: Yes** |
| **Is the dataset is in encoded format of PCA values: Yes** |
| **Is it essential to pre-process the dataset for the case study: Yes**  **If Yes, how you want to preprocess? Give details:**   1. **Handling Missing Values :** Check for missing data and apply appropriate strategies (removal or imputation). 2. **Feature Selection :** Select only the most relevant features to reduce complexity and improve performance. 3. **Handling Imbalanced Data (SMOTE) :** Apply Synthetic Minority Over-sampling Technique (SMOTE) to balance the dataset and prevent bias. 4. **Splitting Data (80% Training, 20% Testing) :** Divide the dataset into training (80%) and testing (20%) sets to evaluate model performance on unseen data**.** 5. **Feature Scaling (Standardization) :** Use StandardScaler to normalize numerical features, ensuring they have similar ranges for better model efficiency**.**   **These preprocessing steps improve model accuracy, prevent bias, and help the model generalize well to new data.** |
| **List out the possible opportunities for analysis on this dataset based on the available features**   1. **Customer Churn Prediction → Predict whether a customer will leave the bank based on their attributes.** 2. **Customer Segmentation → Group customers based on demographics and behaviors for targeted marketing.** 3. **Feature Importance Analysis → Identify key factors influencing customer churn.** 4. **Correlation Analysis → Examine relationships between features to understand churn drivers.** 5. **Customer Retention Strategy → Develop strategies to retain high-risk customers.** 6. **Impact of Geography on Churn → Analyze how customer location affects churn rates.** 7. **Product Usage Analysis → Study how the number of products impacts customer retention.** 8. **Customer Lifetime Value (CLV) Prediction → Estimate a customer’s long-term profitability.** 9. **Churn Trend Analysis Over Time → Identify patterns in churn based on tenure and age.** |

|  |
| --- |
| **Title of the Case Study:** Customer Churn Prediction and Analysis |
| **List of Objectives:**   * Develop a machine learning model to predict whether a customer will exit the bank. * Analyze key factors such as credit score, balance, and activity level that influence churn. * Handle class imbalance using techniques like SMOTE for better model performance. * Optimize model accuracy by training and evaluating Logistic Regression, XGBOOST. * Segment customers based on demographics and behavior for targeted retention strategies. * Perform feature correlation analysis to understand relationships between variables and their impact on churn. * Identify churn trends based on age, tenure, and geography to uncover patterns. * Develop data-driven retention strategies to reduce churn and improve customer loyalty. * Visualize findings using confusion matrices, heatmaps, and feature importance plots. * Provide actionable insights for banks to enhance customer experience and minimize churn rates |
| **Approach: What features are going to be considered, processed, or feature-engineered to derive a specific outcome after applying one or more models?**   * CreditScore * Age * Tenure * Balance * NumOfProducts * HasCrCard * IsActiveMember * EstimatedSalary * Geography (Encoded) * Gender (Encoded) |

|  |
| --- |
| **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 Collection:**  Load the dataset and understand its structure, features, and target variable.  **2. Data Preprocessing**   * Handle missing values (if any). * Encode categorical variables (e.g., Geography, Gender). * Scale numerical features using StandardScaler. * Address class imbalance using SMOTE.   3. **Feature Selection:**   * Select relevant features: CreditScore, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary. * Perform correlation analysis to understand feature relationships.   **4. Feature Engineering:**   * Create interaction features (e.g., CreditScore × IsActiveMember). * Convert continuous variables into categorical bins (e.g., Age groups).   **5.Splitting Data:**  Divide the dataset into 80% training data and 20% testing data.  **6. Model Selection**   * Choose machine learning models: Logistic Regression, XGBoost. * Perform hyperparameter tuning for optimized model performance.   7. **Model Training**   * Train models on preprocessed training data. * Use RandomizedSearchCV for hyperparameter optimization.   **8. Model Testing**   * Evaluate the models on test data. * Compare predictions with actual values.   **9. Model Evaluation Metrics**   * Accuracy : Measure overall correct predictions. * Precision, Recall, F1-Score : Assess model effectiveness in predicting churn. * Confusion Matrix : Visualize true positives, false positives, etc. * Feature Importance Analysis : 1 Identify key drivers of churn**.**   **10. Visualization & Insights**   * Plot confusion matrices, feature importance, heatmaps to interpret results. |
|  |

**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 |
| Step-1 | Loading Dataset with name, display its descriptive information |
| Description | The Churn\_Modelling dataset is loaded and its descriptive information, including missing values and duplicates, is displayed. |
| Code | import pandas as pd  # Load dataset  df = pd.read\_csv("/content/Churn\_Modelling.csv") # Update with correct path  # Display dataset info  print("Dataset Information:")  print(df.info())  # Check for missing values  print("\nMissing Values:")  print(df.isnull().sum())  # Remove duplicate records  df.drop\_duplicates(inplace=True) |
| Result | The dataset is successfully loaded.   * Information about the dataset is displayed, such as the number of columns, data types, and missing values. * Missing values are checked, and duplicate records (if any) are removed. |
| Description about results in detailed way | We noticed that data is   * The dataset contains multiple features like CreditScore, Age, Balance, and Exited (target variable). * No missing values were found. * The dataset is clean and ready for further preprocessing. |
|  |  |

|  |  |
| --- | --- |
| Task | Data Pre-processing |
|  |  |
| Step -2 | Dropping Parameters / Columns |
| Description | The Churn\_Modelling dataset is pre-processed by dropping unnecessary columns. We noticed that some columns, such as customer-specific identifiers and irrelevant data, do not contribute to the prediction model. |
| Code | df.drop(columns=["RowNumber", "CustomerId", "Surname"], inplace=True) |
| Result | * The columns **RowNumber, CustomerId, and Surname** have been removed. * The dataset is now more refined and contains only relevant features for model training. |
| Description about results in detailed way | * We noticed that the dataset had unnecessary columns such as **RowNumber, CustomerId, and Surname**, which do not provide valuable information for predicting customer churn. *  So, we dropped these columns to enhance model efficiency and reduce computational complexity. |

|  |  |
| --- | --- |
| Task | Data Pre-processing |
| Step | Splitting Dataset and Training Model |
| Description | The dataset is split into training and testing sets to train the model. We use an 80-20 split, and SMOTE is applied to handle class imbalance. The logistic regression model is trained using the best hyperparameters. |
| Code | X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_poly, y, test\_size=0.2, random\_state=42, stratify=y) # Apply SMOTE to balance classes in training data smote = SMOTE(random\_state=42) X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)  scaler = StandardScaler() X\_train\_resampled = scaler.fit\_transform(X\_train\_resampled) X\_test = scaler.transform(X\_test) |
| Result | Missing Values:  RowNumber 0  CustomerId 0  Surname 0  CreditScore 0  Geography 0  Gender 0  Age 0  Tenure 0  Balance 0  NumOfProducts 0  HasCrCard 0  IsActiveMember 0  EstimatedSalary 0  Exited 0 |
| Description about results in detailed way | The first plot shows the original class distribution, where the dataset is highly imbalanced, with significantly more "Not Exited" (0) cases than "Exited" (1) cases. The second plot demonstrates the effect of resampling (such as SMOTE), which balances the dataset by making the number of samples in both classes equal, improving model performance on minority class predictions. |

|  |  |  |  |
| --- | --- | --- | --- |
| Task | |  | | --- | |  |  |  | | --- | | **Model Evaluation** | |
| Step -4 | Calculating Evaluation Metrics - Accuracy |
| Description | Churn\_Modelling.csv dataset is pre-processed with LOGISTIC REGRESSION , We noticed that Model performance is evaluated using accuracy, precision, recall, F1-score, and a classification report. |
| Code | # Predictions  log\_reg\_train\_pred = log\_reg.best\_estimator\_.predict(X\_train\_resampled)  log\_reg\_test\_pred = log\_reg.best\_estimator\_.predict(X\_test)  # Function to print evaluation metrics  def evaluate\_model(name, y\_true\_train, y\_pred\_train, y\_true\_test, y\_pred\_test):  print(f"\n{name} Evaluation Metrics:")  print("Test Set:")  print("Accuracy:", accuracy\_score(y\_true\_test, y\_pred\_test))  print("Precision:", precision\_score(y\_true\_test, y\_pred\_test))  print("Recall:", recall\_score(y\_true\_test, y\_pred\_test))  print("F1 Score:", f1\_score(y\_true\_test, y\_pred\_test))  print("\nClassification Report (Test Set):\n", classification\_report(y\_true\_test, y\_pred\_test))  # Evaluate Logistic Regression  evaluate\_model("Logistic Regression", y\_train\_resampled, log\_reg\_train\_pred, y\_test, log\_reg\_test\_pred) |
| Result |  |
| Description about results in detailed way | The evaluation metrics indicate the performance of the logistic regression model on the test dataset:   * **Accuracy (71.95%)** suggests that the model correctly predicts customer churn in approximately 72% of cases. * **Precision (39.09%)** indicates that out of all the predicted churn cases, only 39% were actually correct, showing a high false positive rate. * **Recall (67.81%)** reflects that the model captures around 68% of actual churn cases, meaning it successfully identifies most churners but at the cost of some false positives. * **F1 Score (49.59%)** is a balance between precision and recall, showing that the model struggles with predicting churn cases accurately. |

|  |  |  |  |
| --- | --- | --- | --- |
| Task | |  | | --- | |  |  |  | | --- | | **Visualization** | |
| Step -4 | Visualization - Confusion Matrix |
| Description | The confusion matrix is plotted to visualize the model’s performance in classifying churn vs. non-churn customers. It helps analyze false positives and false negatives, providing insights into model errors. |
| Code | #  evaluate\_model("Logistic Regression", y\_train\_resampled, log\_reg\_train\_pred, y\_test, log\_reg\_test\_pred)  # Confusion Matrices  log\_reg\_cm\_test = confusion\_matrix(y\_test, log\_reg\_test\_pred)  # Function to plot confusion matrix  def plot\_confusion\_matrix(conf\_matrix, title):      plt.figure(figsize=(6, 5))      sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="coolwarm", linewidths=1, linecolor="black")      plt.xlabel("Predicted")      plt.ylabel("Actual")      plt.title(title)      plt.show()  # Plot confusion matrix  plot\_confusion\_matrix(log\_reg\_cm\_test, "Confusion Matrix - Logistic Regression (Test)" |
| Result |  |
| Description about results in detailed way | The confusion matrix shows that the model correctly predicted 1,163 non-churn cases and 276 churn cases. However, it misclassified 430 non-churn cases as churn (false positives) and 131 churn cases as non-churn (false negatives), indicating areas for improvement in recall and precision. |

|  |  |  |  |
| --- | --- | --- | --- |
| Task | |  | | --- | |  |  |  | | --- | | **Model Evaluation** | |
| Step -4 | Calculating Evaluation Metrics - Accuracy |
| Description | Churn\_Modelling.csv dataset is pre-processed with **RANDOM FOREST**, We noticed that Model performance is evaluated using accuracy, precision, recall, F1-score, and a classification report. |
| Code | # Scale numerical features  scaler = StandardScaler()  X\_train = scaler.fit\_transform(X\_train)  X\_test = scaler.transform(X\_test)  xgb\_model = XGBClassifier(      tree\_method='hist',  # Optimized histogram-based algorithm for speed      use\_label\_encoder=False,  # Disables old label encoding warnings      eval\_metric='logloss',  # Logarithmic loss function for binary classification      n\_jobs=-1      # Uses all available CPU cores for parallel computation  )  xgb\_model.fit(X\_train, y\_train)  # train the model  # Make predictions  y\_pred = xgb\_model.predict(X\_test)  # Apply SMOTE for balancing the dataset  smote = SMOTE(random\_state=42)  X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)  # Evaluate model performance  accuracy = accuracy\_score(y\_test, y\_pred)  print(f"Accuracy: {accuracy:.4f}")  print("Classification Report:")  print(classification\_report(y\_test, y\_pred)) |
| Result |  |
| Description about results in detailed way | The evaluation metrics indicate the performance of the logistic regression model The model achieved an accuracy of **84.7%**, with class 0 (non-event) having higher precision and recall compared to class 1 (event). |

|  |  |  |  |
| --- | --- | --- | --- |
| Task | |  | | --- | |  |  |  | | --- | | **Visualization** | |
| Step -4 | Visualization - Confusion Matrix |
| Description | The confusion matrix is plotted to visualize the model’s performance in classifying churn vs. non-churn customers. It helps analyze false positives and false negatives, providing insights into model errors. |
| Code | conf\_matrix\_test = confusion\_matrix(y\_test, y\_pred)  def plot\_confusion\_matrix(conf\_matrix, title):      plt.figure(figsize=(6, 5))      sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="coolwarm", linewidths=1, linecolor="black")      plt.xlabel("Predicted")      plt.ylabel("Actual")      plt.title(title)      plt.show()  plot\_confusion\_matrix(conf\_matrix\_test, "Confusion Matrix (Test Set)")  # Feature importance visualization  feature\_importance = xgb\_model.feature\_importances\_  feature\_names = X.columns  plt.figure(figsize=(10, 6))  sns.barplot(x=feature\_importance, y=feature\_names, palette="coolwarm")  plt.xlabel("Feature Importance Score")  plt.ylabel("Features")  plt.title("Feature Importance from XGBoost")  plt.show()  # Heatmap of feature correlations  plt.figure(figsize=(10, 6))  sns.heatmap(df[X.columns.tolist() + ["Exited"]].corr(), annot=True, cmap="coolwarm", fmt=".2f", linewidths=1, linecolor="black")  plt.title("Feature Correlation Heatmap")  plt.show() |
| Result |  |
| Description about results in detailed way | The confusion matrix shows that the model correctly classified **1506** true negatives and 87 true positives, while misclassifying **219** false positives and **188** false negatives. The relatively lower false negatives indicate that the model performs reasonably well but can still be improved for better recall. |

|  |  |  |  |
| --- | --- | --- | --- |
| Task | |  | | --- | |  |  |  | | --- | | **Visualization** | |
| Step -4 | Visualization - Confusion Matrix |
| Description | Churn\_Modelling.csv dataset is pre-processed by handling missing values, encoding categorical variables, scaling numerical features, removing outliers, and applying PCA for dimensionality reduction. Additionally, SMOTE was used to balance the dataset, and XGBoost with hyperparameter tuning was implemented to improve model accuracy. |
| Code | **Step-1: Import necessary libraries**  import pandas as pd import numpy as np import seaborn as sns import matplotlib.pyplot as plt from sklearn.model\_selection import train\_test\_split, GridSearchCV from sklearn.preprocessing import StandardScaler, LabelEncoder, RobustScaler from xgboost import XGBClassifier from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix from imblearn.over\_sampling import SMOTE from sklearn.decomposition import PCA  **Load the dataset**  df = pd.read\_csv("Churn\_Modelling.csv")  **Step 3: Handle missing values**  df.dropna(inplace=True)  **Step 4: Outlier removal using IQR**  numerical\_cols = ['CreditScore', 'Age', 'Balance', 'EstimatedSalary'] df\_clean = df.copy() for col in numerical\_cols: Q1 = df\_clean[col].quantile(0.25) Q3 = df\_clean[col].quantile(0.75) IQR = Q3 - Q1 lower\_bound = Q1 - 1.5 \* IQR upper\_bound = Q3 + 1.5 \* IQR df\_clean = df\_clean[(df\_clean[col] >= lower\_bound) & (df\_clean[col] <= upper\_bound)]  **Step 5: Encode categorical variables**  categorical\_cols = ['Gender', 'Geography'] label\_encoders = {} for col in categorical\_cols: le = LabelEncoder() df\_clean[col] = le.fit\_transform(df\_clean[col]) label\_encoders[col] = le  **Step 6: Define features (X) and target variable (y)**  X = df\_clean.drop(columns=['Exited', 'CustomerId', 'Surname', 'RowNumber'], errors='ignore') y = df\_clean['Exited']  **Step 7: Split dataset into training and testing sets**  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)  **Step 8: Apply SMOTE to balance classes**  smote = SMOTE(sampling\_strategy=0.6, random\_state=42, k\_neighbors=3) X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)  **Step 9: Scale data using RobustScaler to handle outliers**  scaler = RobustScaler() X\_train\_resampled = scaler.fit\_transform(X\_train\_resampled) X\_test = scaler.transform(X\_test)  **Step 10: Apply PCA for dimensionality reduction (retaining 95% variance)**  pca = PCA(n\_components=0.95) X\_train\_resampled = pca.fit\_transform(X\_train\_resampled) X\_test = pca.transform(X\_test)  **Step 11: Define hyperparameter grid for XGBoost**  param\_grid = { 'n\_estimators': [100, 200, 300], 'learning\_rate': [0.01, 0.05, 0.1], 'max\_depth': [3, 5, 7], 'subsample': [0.8, 1], 'colsample\_bytree': [0.8, 1] }  **Step 12: Perform hyperparameter tuning using GridSearchCV**  xgb = XGBClassifier(tree\_method='hist', use\_label\_encoder=False, eval\_metric='logloss', n\_jobs=-1) grid\_search = GridSearchCV(xgb, param\_grid, cv=3, scoring='accuracy', n\_jobs=-1, verbose=1) grid\_search.fit(X\_train\_resampled, y\_train\_resampled)  **Step 13: Train the best model**  best\_xgb = grid\_search.best\_estimator\_ y\_pred = best\_xgb.predict(X\_test)  **Step 14: Evaluate model performance**  accuracy = accuracy\_score(y\_test, y\_pred) print(f"Optimized Accuracy: {accuracy:.4f}") print("Classification Report:") print(classification\_report(y\_test, y\_pred))  **Step 15: Confusion matrix visualization**  conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(6, 5)) sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="coolwarm", linewidths=1, linecolor="black") plt.xlabel("Predicted") plt.ylabel("Actual") plt.title("Confusion Matrix") plt.show()  **Step 16: Feature importance visualization**  feature\_importance = best\_xgb.feature\_importances\_ feature\_names = X.columns[:len(feature\_importance)] # Match with reduced PCA features  plt.figure(figsize=(10, 6)) sns.barplot(x=feature\_importance, y=feature\_names, palette="coolwarm") plt.xlabel("Feature Importance Score") plt.ylabel("Features") plt.title("Feature Importance from Optimized XGBoost") plt.show() |
| Result |  |
| Description about results in detailed way |  Removing outliers helps improve model accuracy by reducing noise.   It ensures the dataset follows a more normal distribution, improving statistical analysis.   The IQR method effectively identifies extreme values while preserving most of the useful data. |