**Credit Card Fraud Detection**

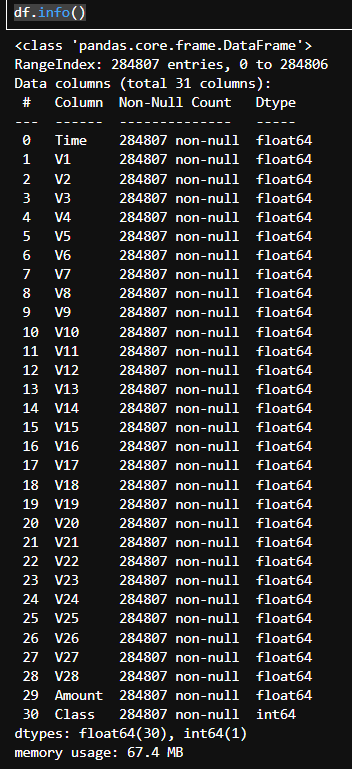
In This project your goal is to detect that customers are not charged for items that they did not purchase.

The first problem is that the data are imbalanced so the traditional techniques will be useless so we hade to find a way to make the data ready and suitable

The way I used called  Linear Discriminant Analysis AKA as **LDA** which is supervised classification problem that helps separate two or more classes by converting higher-dimensional data space into a lower-dimensional space

**Data Preprocessing**

As we can see the data don’t have nulls which is good



1-Standardize the Amount column

2- Drop Time column

3- Remove duplicates

When I see the class count I make sure that I’m dealing with imbalanced data so I hade to make a move to fix this

The main idea is **Dimensionality Reduction**

Step 1: Separate features and target variable

X = df.drop('Class', axis=1) # Features

y = df['Class'] # Target variable

Step 2: Split Data into Training and Testing Sets

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

Step 3: Standardize features ensures that all features have the same scale.

* Standardization is essential for LDA as it ensures that all features have the same scale.
* This prevents features with larger scales from dominating the principal components

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

Step 4:Address Class Imbalance with SMOTE

a generating samples using SMOTE applyed SMOTE for the lake and imbalanced data

# Apply SMOTE to address class imbalance

print("\n Imbalance with SMOTE:")

smote = SMOTE(random\_state=0)

X\_train\_over, y\_train\_over = smote.fit\_resample(X\_train, y\_train)

# Display information before and after SMOTE

print("Before SMOTE: ", X\_train.shape, y\_train.shape)

print("After SMOTE: ", X\_train\_over.shape, y\_train\_over.shape)

print("After SMOTE (0,1): ", pd.Series(y\_train\_over).value\_counts())

From the output you can see that we have the same number for both classes

Which show us the power of SMOTE

Imbalance with SMOTE:

Before SMOTE: (192964, 29) (192964,)

After SMOTE: (385254, 29) (385254,)

After SMOTE (0,1): Class

0 192627

1 192627

**apply LDA**

# Perform LDA for dimensionality reduction

lda = LDA(n\_components=1)

X\_train\_lda = lda.fit\_transform(X\_train\_over, y\_train\_over)

X\_test\_lda = lda.transform(X\_test)

 Train SVM Model with Hyperparameter Tuning[¶](https://www.kaggle.com/code/ayanabil11/lda-svm#Step-1:-Train-SVM-Model-with-Hyperparameter-Tuning)

* Support Vector Machine (SVM) is a powerful classification algorithm that works well for both linear and non-linear datasets.
* Hyperparameters like kernel type, C (regularization parameter), and gamma influence the model's performance.
* For brevity, we'll demonstrate training with default hyperparameters.

# Train SVM Model with Hyperparameter Tuning

svm\_model = SVC(kernel='rbf', C=1, gamma='scale')

svm\_model.fit(X\_train\_over, y\_train\_over)

Make predictions

y\_pred = svm\_model.predict(X\_test)

Evaluate Model Performance

If we did not make the **Dimensionality Reduction** the confusion matrix wont be accurate

# Evaluate Model Performance

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | Predicted 0 | Predicted 1 |
| Actual 0 | 81320 | 1243 |
| Actual 1 | 18 | 118 |

Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 1.00 | 0.98 | 0.99 | 82563 |
| 1 | 0.09 | 0.87 | 0.16 | 136 |
| accuracy |  |  | 0.98 | 82699 |
| macro avg | 0.54 | 0.93 | 0.57 | 82699 |
| weighted avg | 1.00 | 0.98 | 0.99 | 82699 |

Calculate ROC AUC Score

# Calculate ROC AUC Score

roc\_auc = roc\_auc\_score(y\_test, y\_pred)

print("\nROC AUC Score:", roc\_auc)

ROC AUC Score: 0.9262959444160644