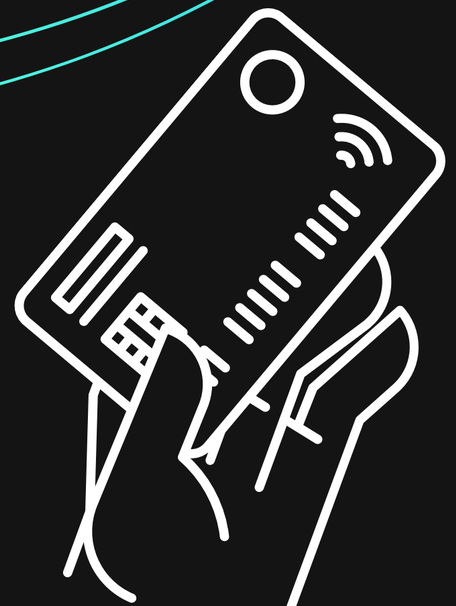
Credit Card Fraud Detection

SC1015 MINI PROJECT
GROUP 11 (FCSD)
ARPITA, CHAVI GOYAL,
SAMRIDDHI



PIPELINE

INTRODUCTION

PRACTICAL MOTIVATION & PROBLEM STATEMENT

EDA

EXPLORATORY DATA ANALYSIS & ANALYTIC VISUALIZATION

ALGORITHMIC OPTIMIZATION

DIMENSION REDUCTION, CLASSIFIERS

MACHINE LEARNING

CLASSIFIERS, CROSS VALIDATION, LEARNING CURVES

CONCLUSION

STATISTICAL INFERENCE

DATASET:

Credit Card Fraud Detection

Anonymized credit card transactions labeled as fraudulent or genuine

SOURCE: KAGGLE DATASETS
CREDIT CARD FRAUD DETECTION BY ULB & ANDREA

Motivation

BUILD MODELS THAT ACCURATELY DETECT FRAUDULENT TRANSACTIONS, HELPING TO PREVENT FINANCIAL LOSSES FOR INDIVIDUALS AND BUSINESSES

Problem Statement

We want to build a machine learning model that can accurately detect fraudulent credit card transactions, thereby preventing unauthorized charges and financial losses

PIPELINE

INTRODUCTION

PRACTICAL MOTIVATION & PROBLEM STATEMENT

EDA

EXPLORATORY DATA ANALYSIS & ANALYTIC VISUALIZATION

ALGORITHMIC OPTIMIZATION

DIMENSION REDUCTION, CLASSIFIERS

MACHINE LEARNING

CLASSIFIERS, MODEL EVALUATION, LEARNING CURVES

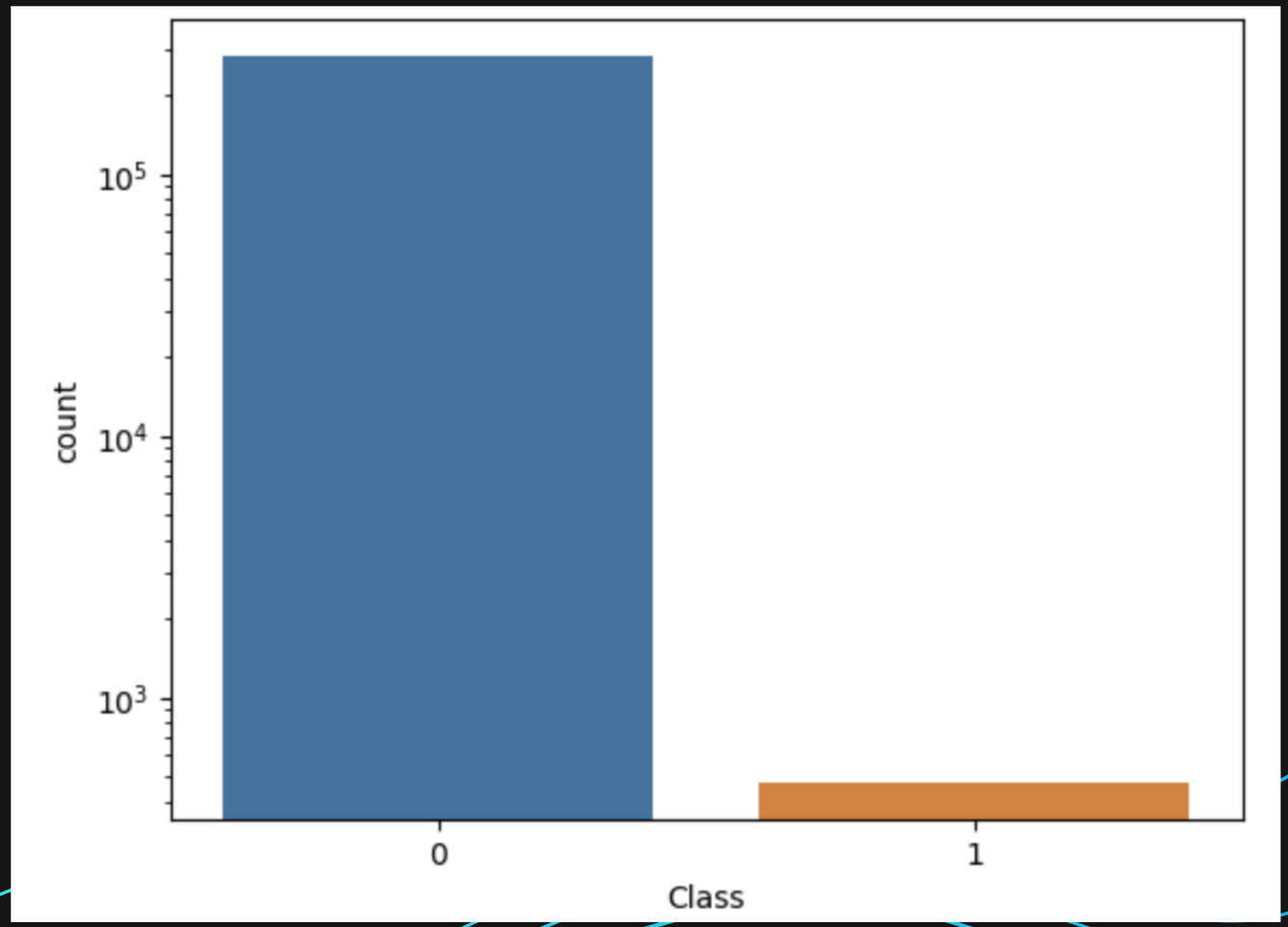
CONCLUSION

STATISTICAL INFERENCE

DATA:

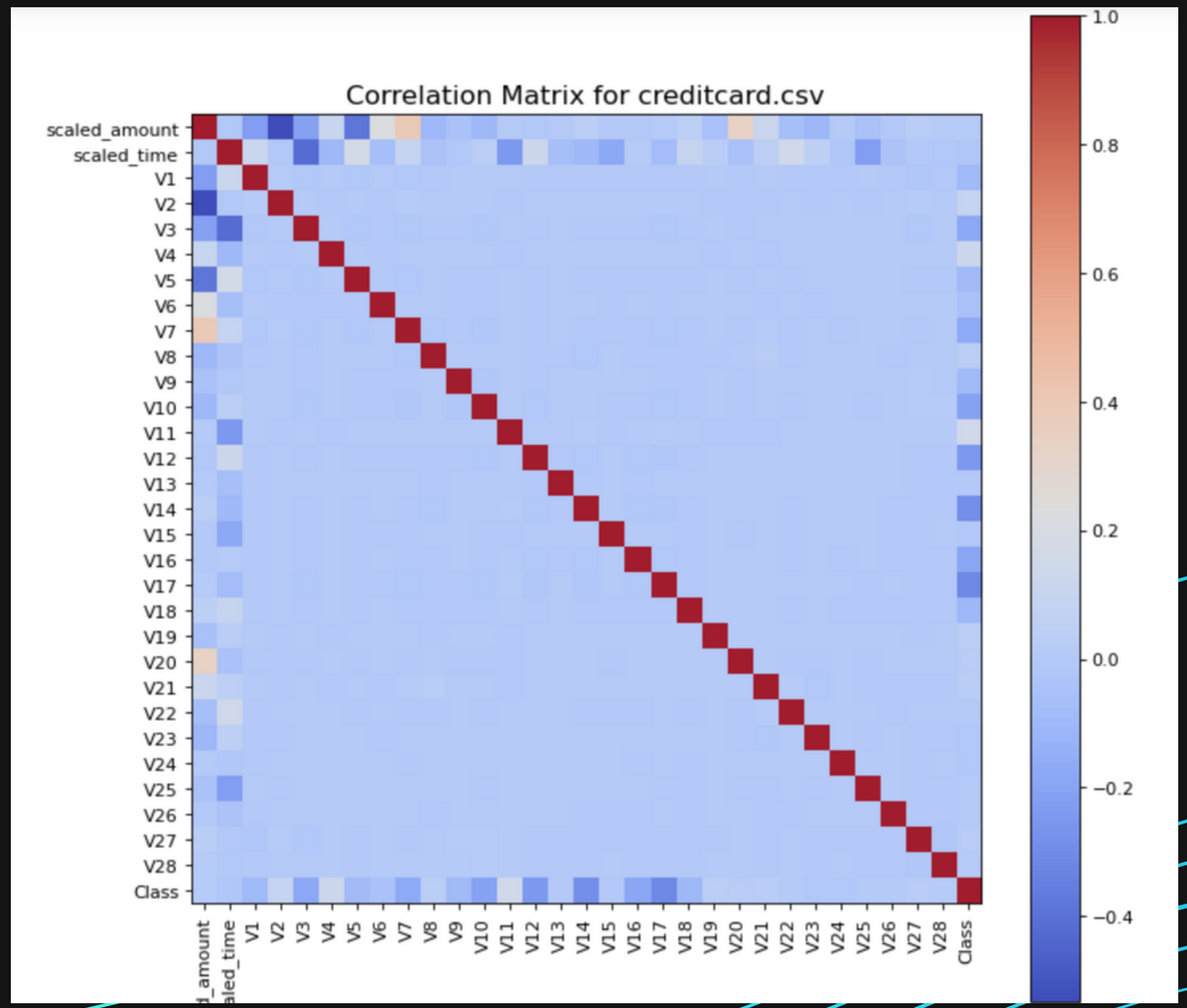
#	Column	Non-Null Count	Dtype
0	Time	284807 non-null	float64
1	V1	284807 non-null	float64
2	V2	284807 non-null	float64
3	V3	284807 non-null	float64
4	V4	284807 non-null	float64
5	V5	284807 non-null	float64
6	V6	284807 non-null	float64
7	V7	284807 non-null	float64
8	V8	284807 non-null	float64
9	V9	284807 non-null	float64
10	V10	284807 non-null	float64
11	V11	284807 non-null	float64
12	V12	284807 non-null	float64
13	V13	284807 non-null	float64
14	V14	284807 non-null	float64
15	V15	284807 non-null	float64
16	V16	284807 non-null	float64
17	V17	284807 non-null	float64
18	V18	284807 non-null	float64
19	V19	284807 non-null	float64
20	V20	284807 non-null	float64
21	V21	284807 non-null	float64
22	V22	284807 non-null	float64
23	V23	284807 non-null	float64
24	V24	284807 non-null	float64
25	V25	284807 non-null	float64
26	V26	284807 non-null	float64
27	V27	284807 non-null	float64
28	V28	284807 non-null	float64
29	Amount	284807 non-null	float64
30	Class	284807 non-null	int64

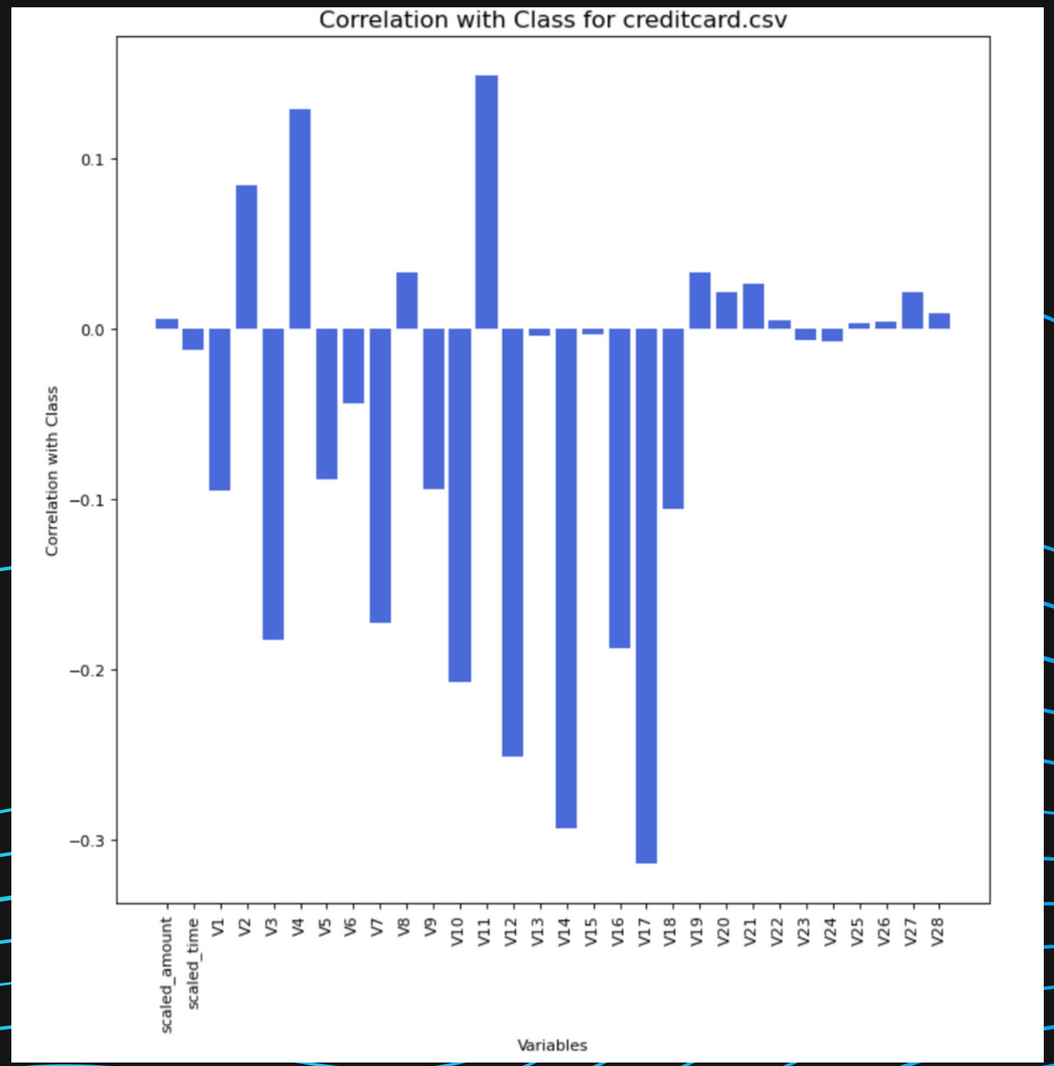
SKEWED



CORRELATION

We select features V17,V14,V12,V10 since they have >0.2 correlation





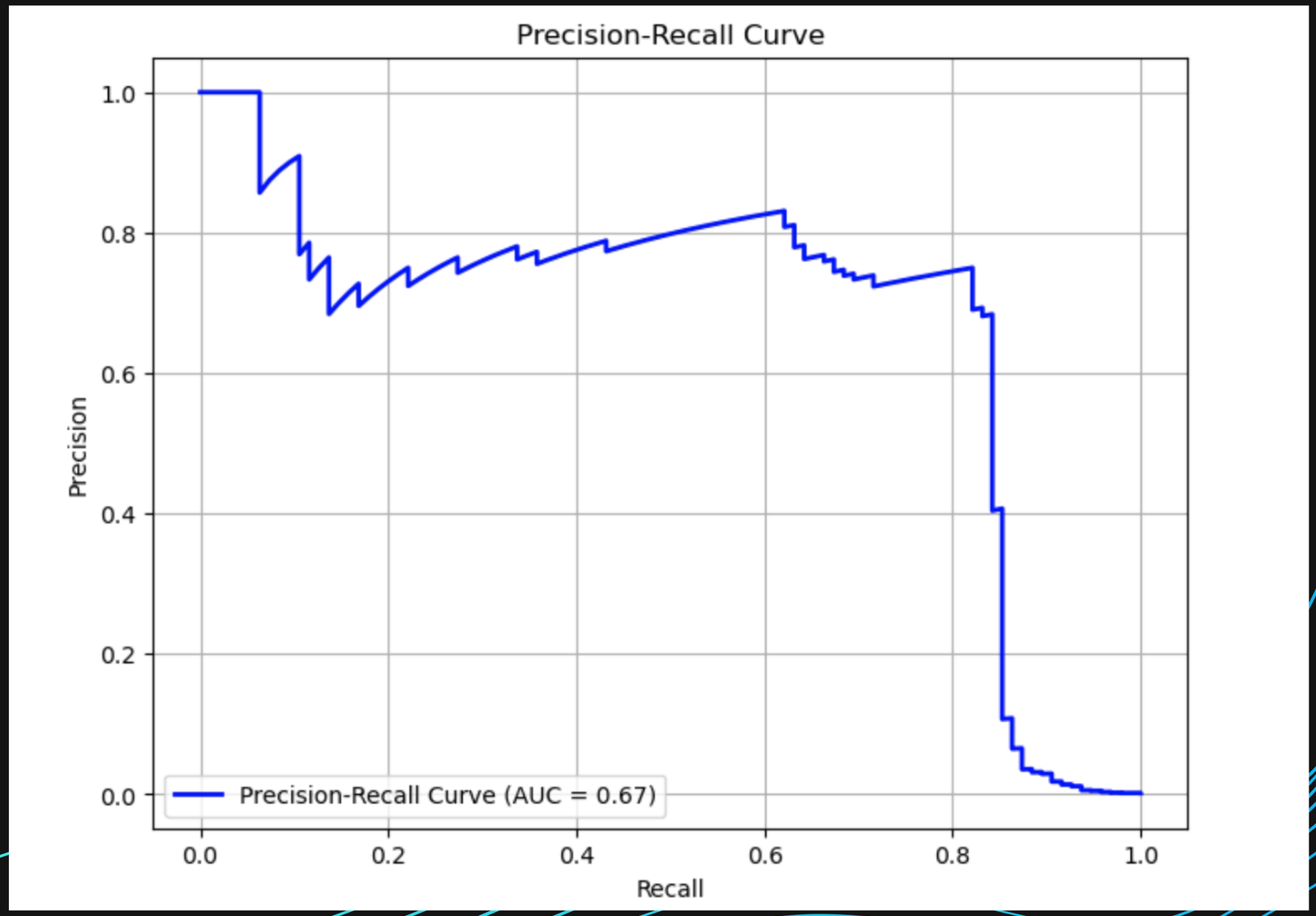
SAMPLING- NEAR MISS AND UNDERSAMPLING

In [30]:

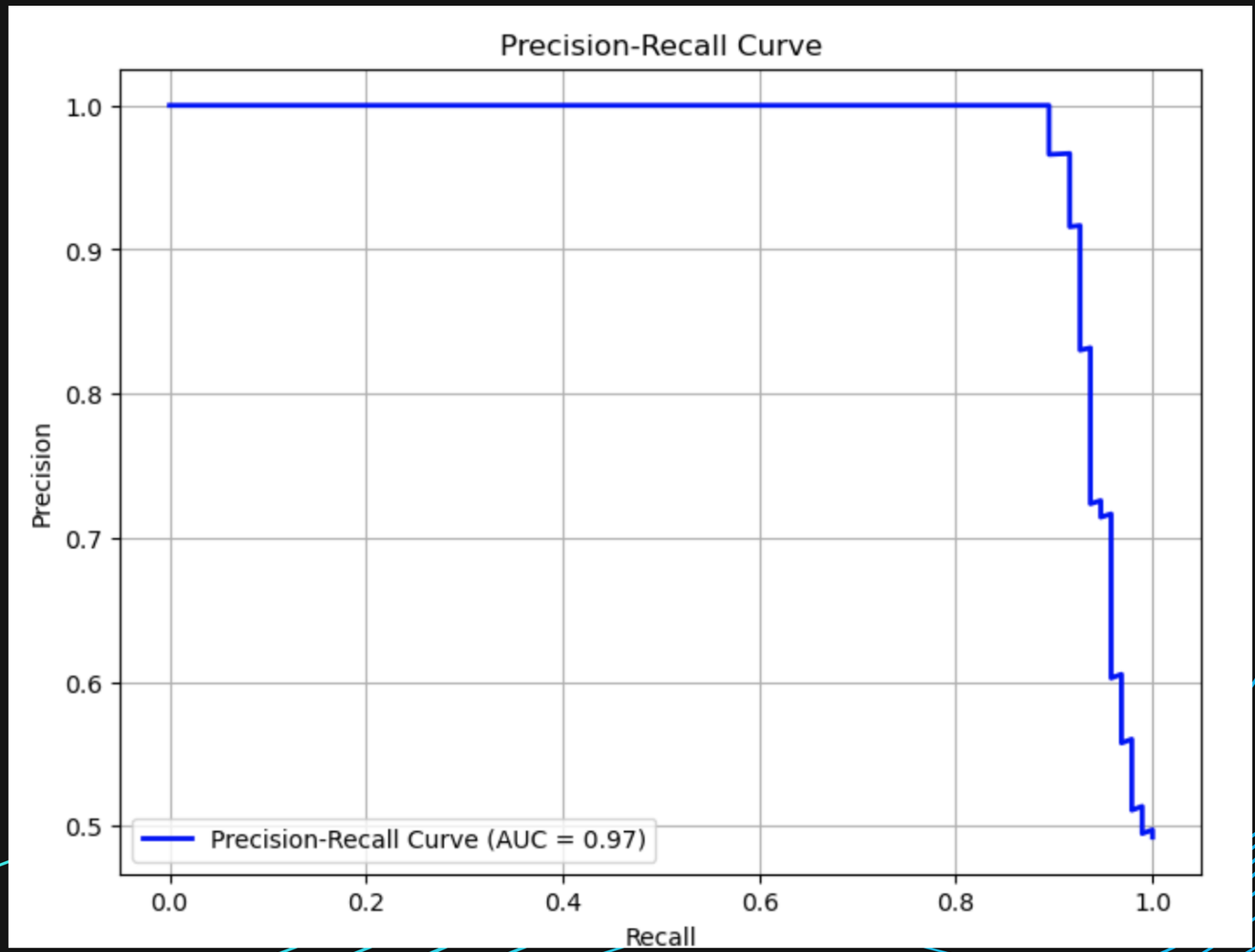
```
import seaborn as sns
import matplotlib.pyplot as plt

print('Distribution of the Classes in the subsample dataset')
print(new_df['Class'].value_counts()/len(new_df))

# Define color palette if needed
colors = ["#0101DF", "#DF0101"]

# Plot the countplot
sns.countplot(x='Class', data=new_df, palette=colors)
plt.title('Equally Distributed Classes', fontsize=14)
plt.show()
```

```
Distribution of the Classes in the subsample dataset
0    0.509845
1    0.490155
Name: Class, dtype: float64
```

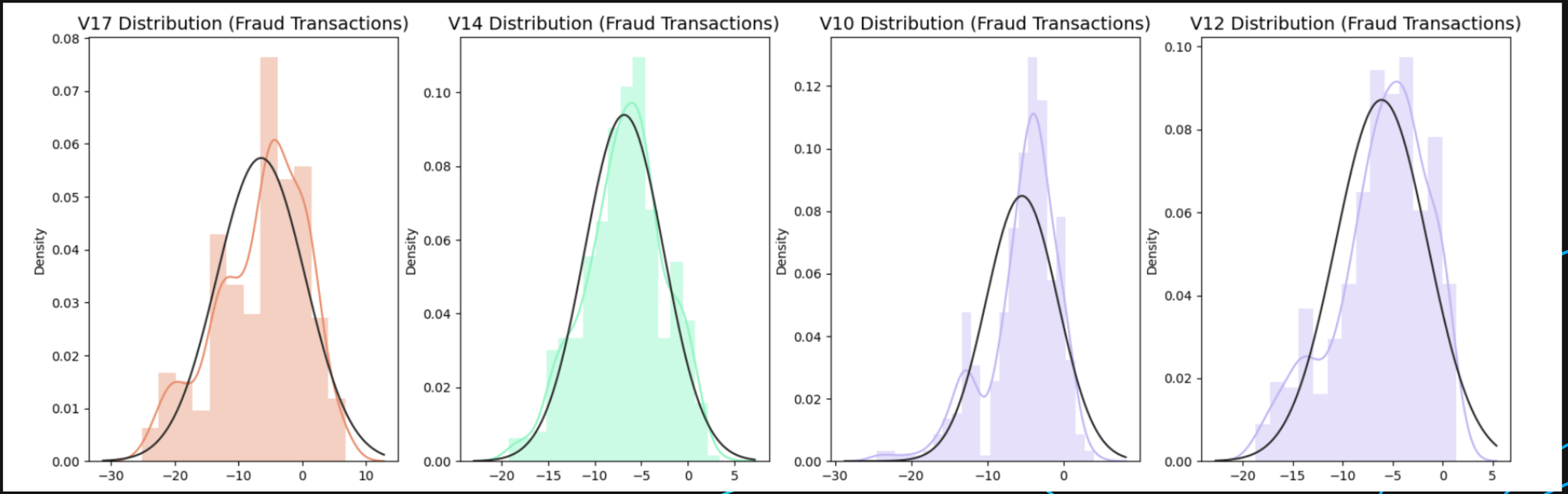
AUC BEFORE UNDER- SAMPLING



AUC AFTER UNDER- SAMPLING

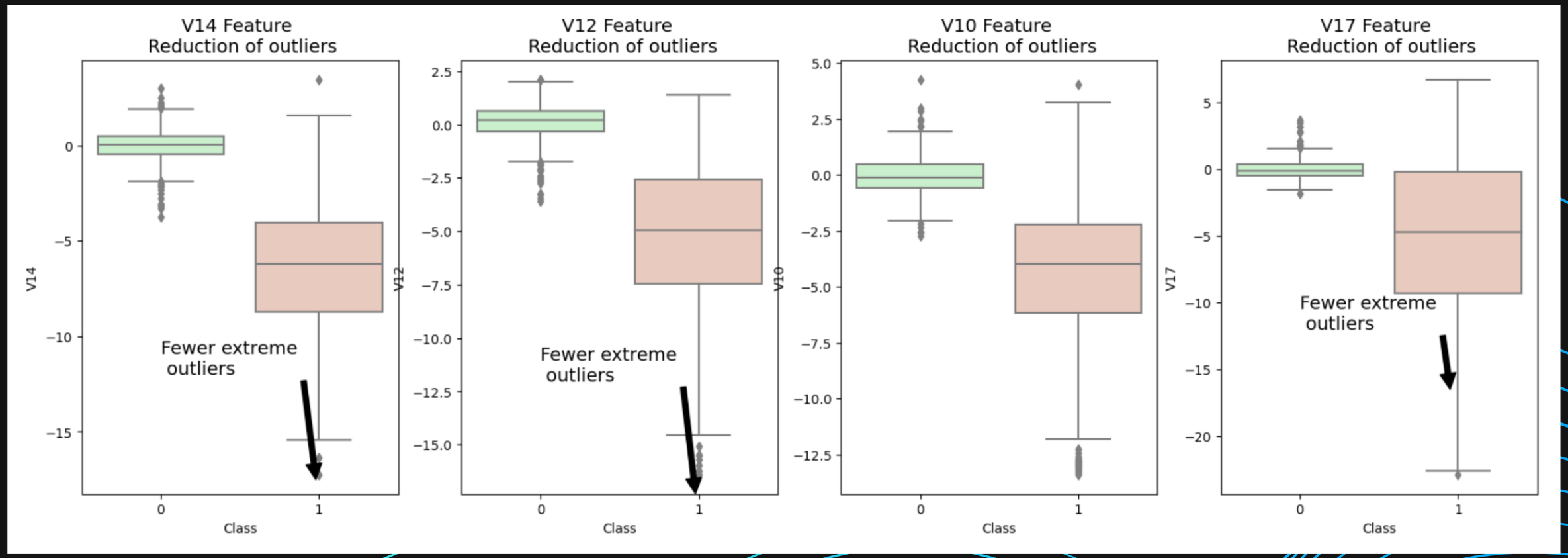


OUTLIERS REMOVED




```
Number of Instances after outliers removal for V12: 950
Number of Instances after outliers removal for V10: 918
Number of Instances after outliers removal for V14: 957
Number of Instances after outliers removal for V17: 914
```

OUTLIERS REMOVED



PIPELINE

INTRODUCTION

PRACTICAL MOTIVATION & PROBLEM STATEMENT

EDA

EXPLORATORY DATA ANALYSIS & ANALYTIC VISUALIZATION

ALGORITHMIC OPTIMIZATION

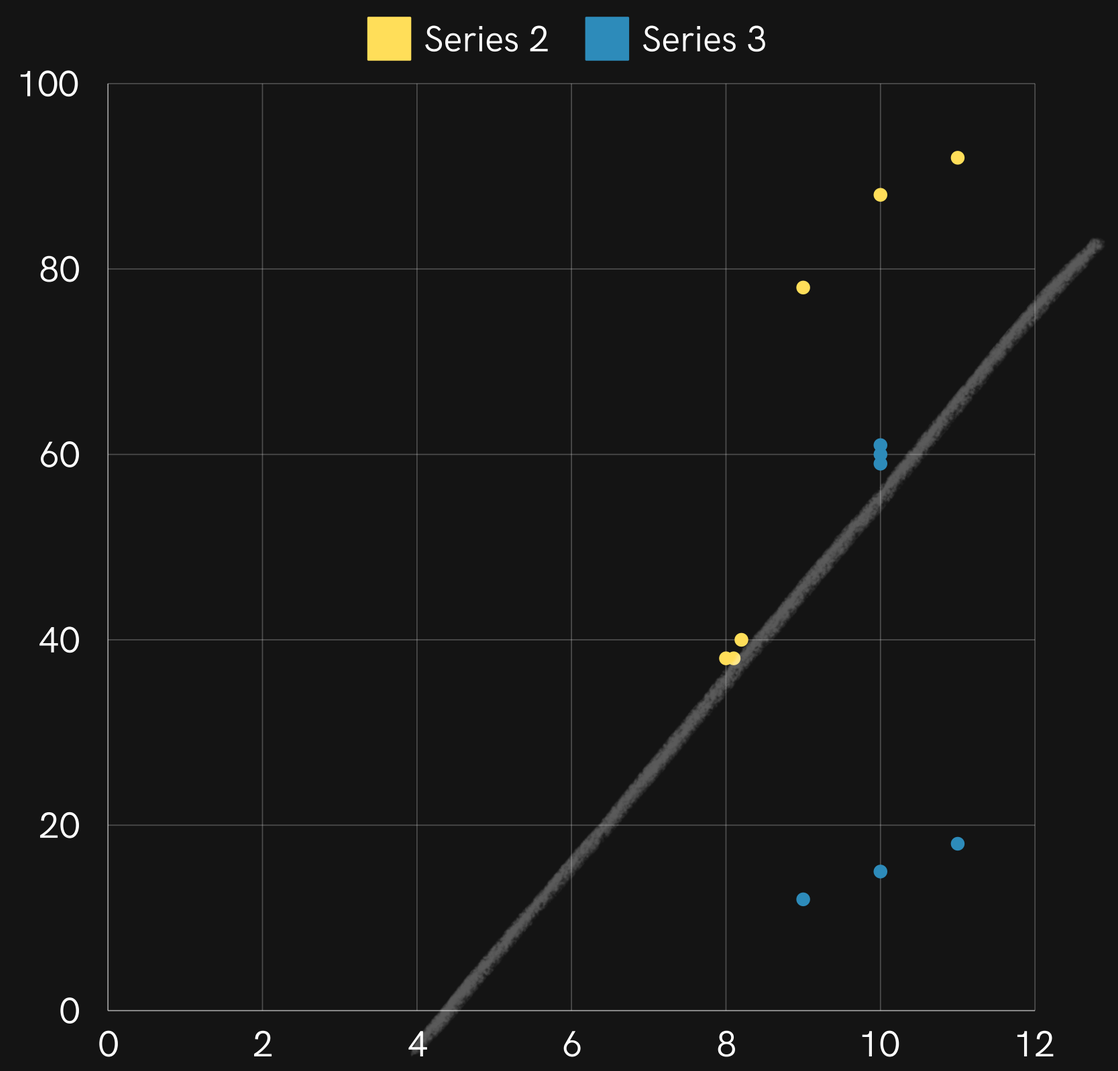
DIMENSION REDUCTION, CLASSIFIERS

MACHINE LEARNING

CLASSIFIERS, MODEL EVALUATION, LEARNING CURVES

CONCLUSION

STATISTICAL INFERENCE

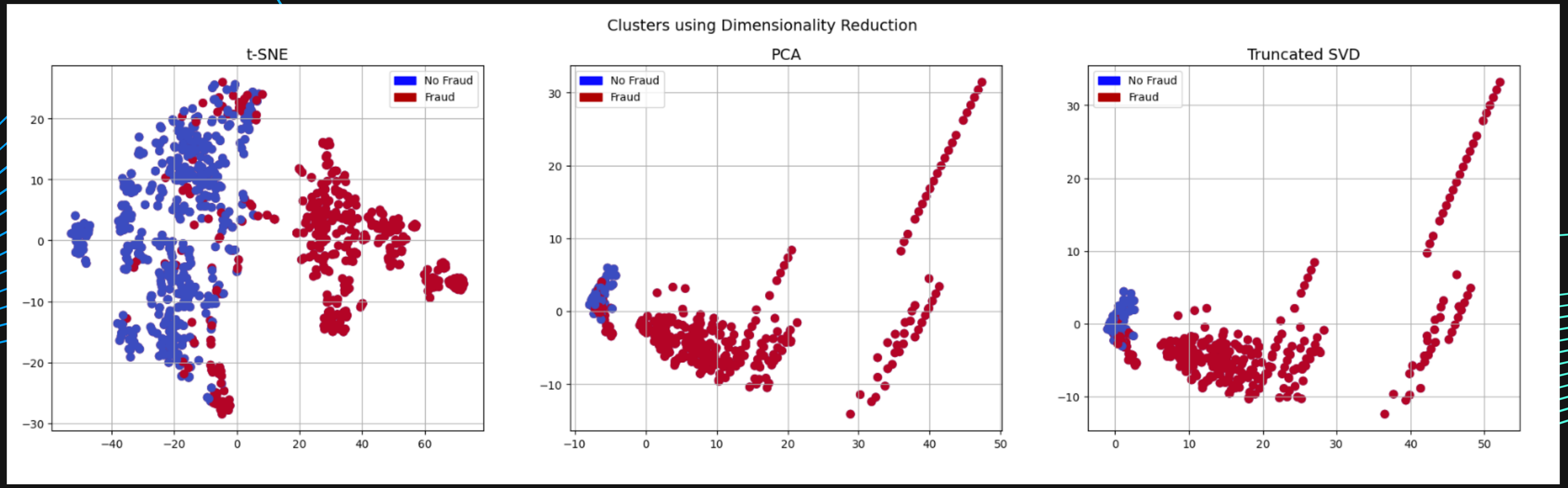


Dimension Reduction

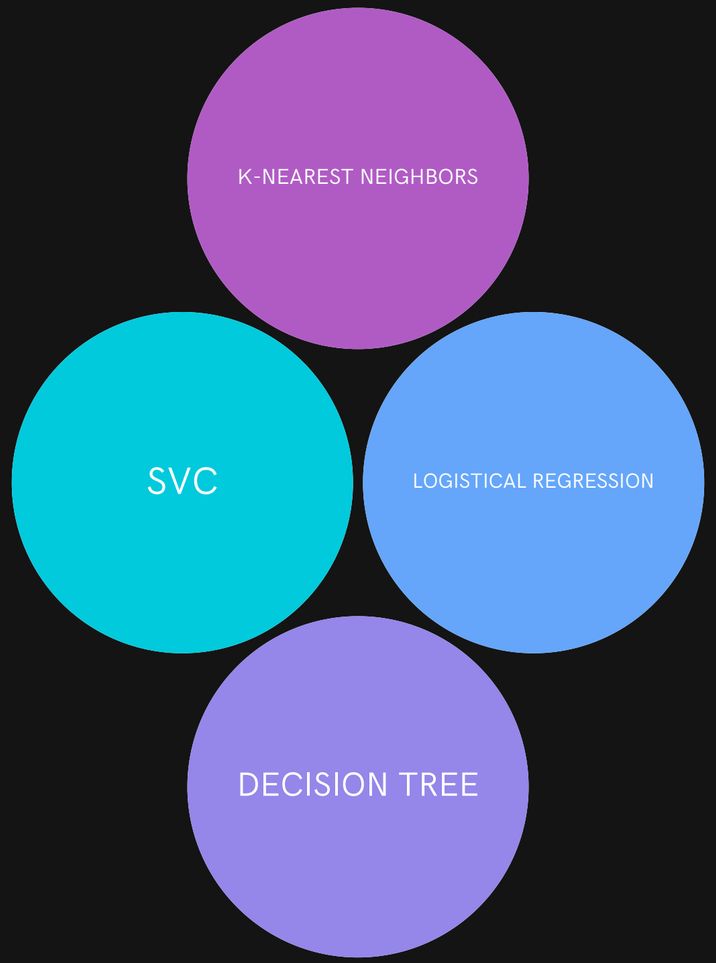
REDUCE DIMENSIONS WHILE PRESERVING CLUSTER IN INFORMATION

Instead of directly projecting the point on the line, they were fitted in order to preserve the clustering

DIMESIONALITY REDUCTION



We Choose four Classifiers:



PIPELINE

INTRODUCTION

PRACTICAL MOTIVATION & PROBLEM STATEMENT

EDA

EXPLORATORY DATA ANALYSIS & ANALYTIC VISUALIZATION

ALGORITHMIC OPTIMIZATION

DIMENSION REDUCTION,CLASSIFIERS

MACHINE LEARNING

CLASSIFIERS, MODEL EVALUATION, LEARNING CURVES

CONCLUSION

STATISTICAL INFERENCE

CROSS VALIDATION

```
Classifier: LogisticRegression has a training score of 94.0 % accuracy score
Classifier: KNeighborsClassifier has a training score of 92.0 % accuracy score
Classifier: SVC has a training score of 93.0 % accuracy score
Classifier: DecisionTreeClassifier has a training score of 91.0 % accuracy score
```

```
Logistic Regression Cross Validation Score:  94.12%
KNearest Neighbors Cross Validation Score 93.3%
Support Vector Classifier Cross Validation Score 93.43%
Decision Tree Classifier Cross Validation Score 91.93%
```

GRIDSEARCH CV

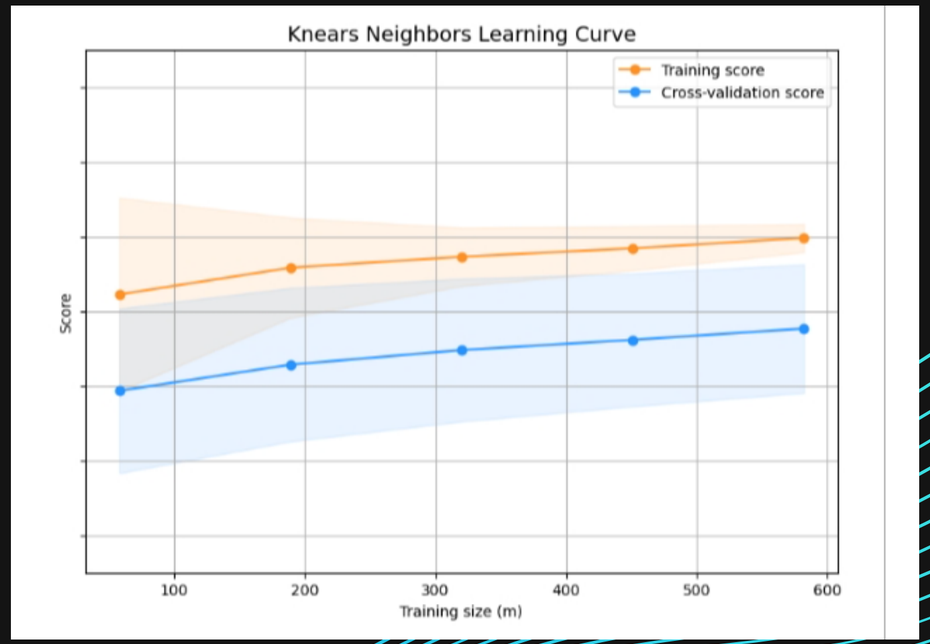
Thus we decide to drop Decision Tree

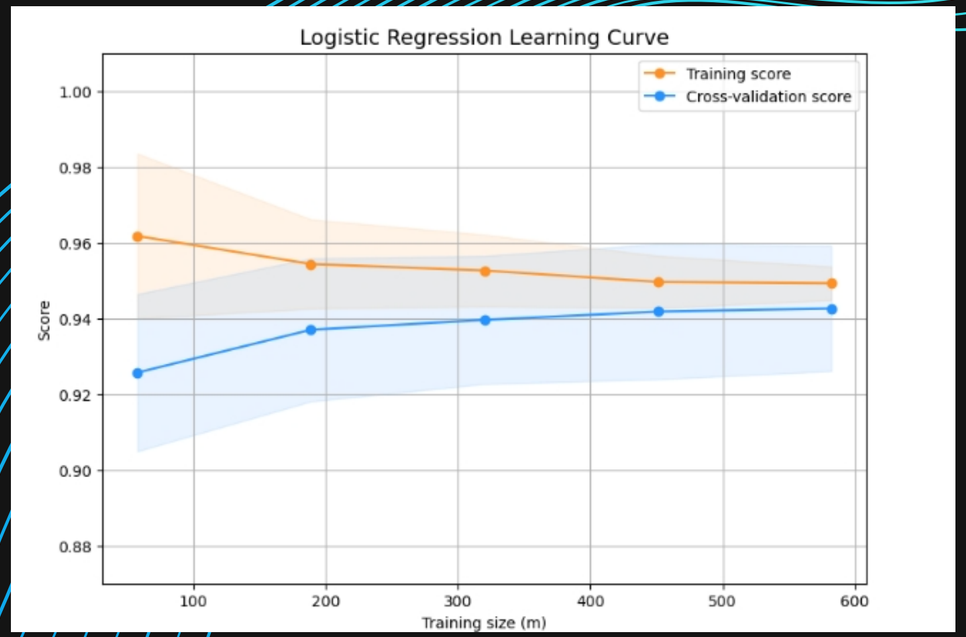
ROC-AUC SCORE

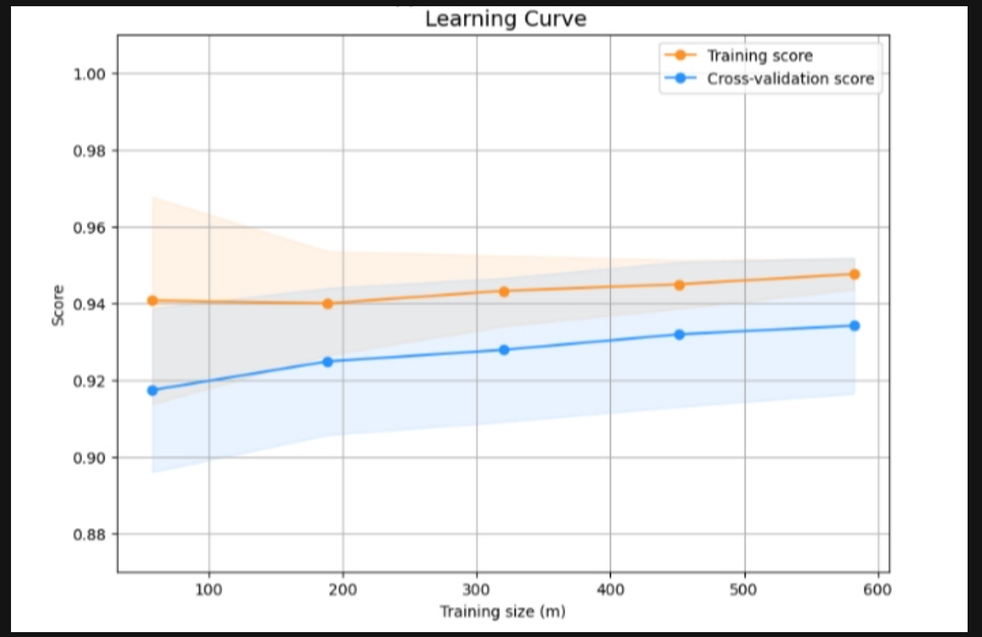
```
Logistic Regression:  0.9737854266278526
KNears Neighbors:  0.9290223080287786
Support Vector Classifier:  0.9701277545664339
Decision Tree Classifier:  0.9145273684369296
```

Thus we choose Logistical Regression

LEARNING CURVES







Predicting test after training on selected features- .96 ROC AUC

```
x_new = new_df[['V10', 'V12', 'V14', 'V17']]
Y_new = new_df['Class']
x_train, x_test, Y_train, Y_test = train_test_split(x_new, Y_new, test_size=0.2, random_state=42)

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score, roc_auc_score

# Step 1: Split into features and target variable
X = df[['V10', 'V12', 'V14', 'V17']]
y = df['Class']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 2: Initialize the logistic regression model
log_reg = LogisticRegression()

# Step 3: Train the logistic regression model
log_reg.fit(x_train, Y_train)

# Step 4: Make predictions on the test data
y_pred = log_reg.predict(X_test)

# Step 5: Calculate precision and ROC-AUC score
precision_pf = precision_score(y_test, y_pred)
roc_auc_pf = roc_auc_score(y_test, log_reg.predict_proba(X_test)[:, 1])  # Extract probabilities for the positive class

print("Precision on test data :", precision_pf)
print("ROC AUC score on test data :", roc_auc_pf)
```

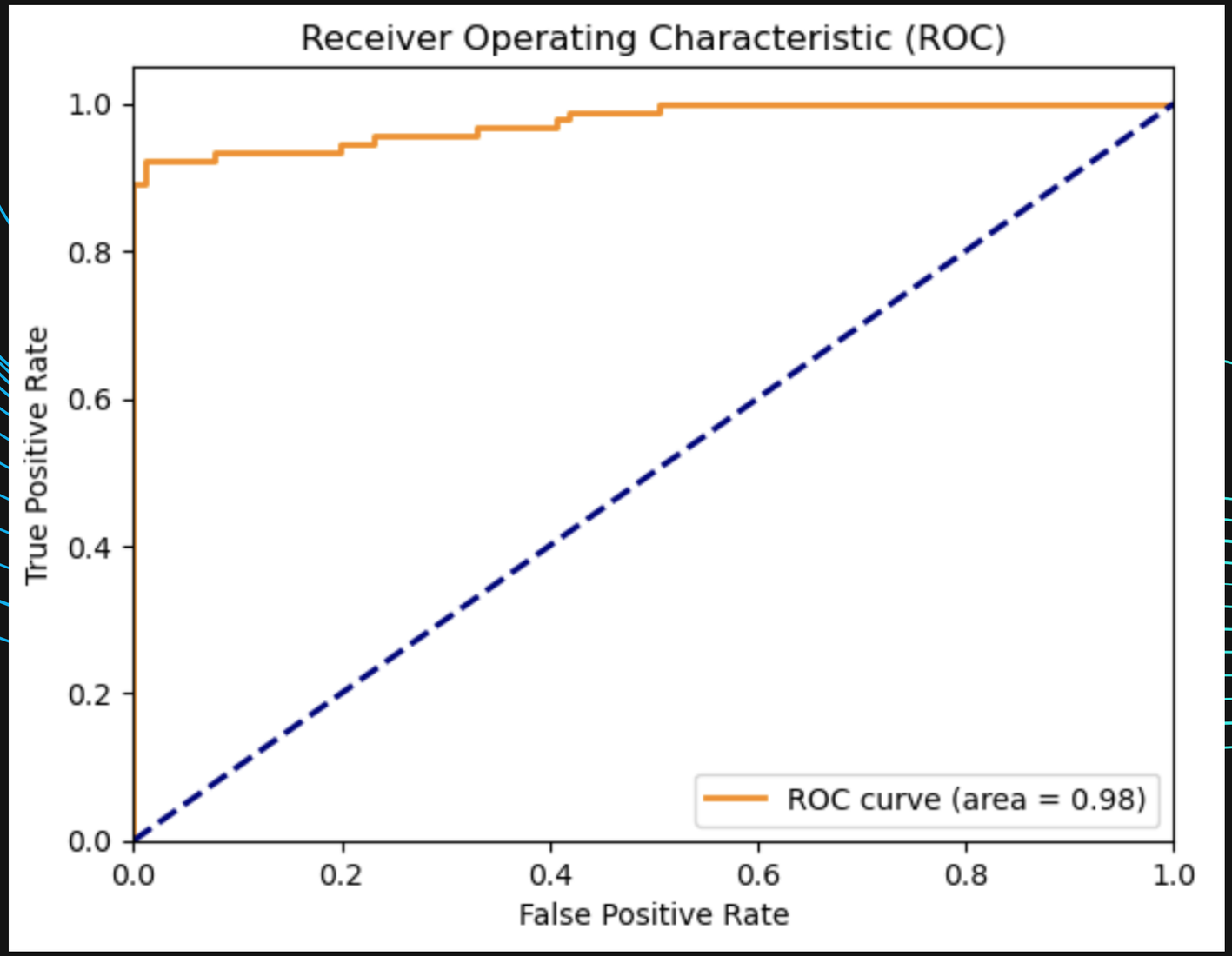
```
Precision on test data : 0.050120772946859904
ROC AUC score on test data : 0.9671138902760039
```

Predicting test after training on all columns-.97 ROC AUC

In [58]:

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score, roc_auc_score
from sklearn.preprocessing import StandardScaler

# Step 1: Split into features and target variable
X = new_df.drop('Class', axis=1)
y = new_df['Class']

# Our data is already scaled we should split our training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 2: Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 3: Train logistic regression model
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

# Step 4: Make predictions on the test data
y_pred = log_reg.predict(X_test)

# Step 5: Calculate precision and ROC-AUC score
precision_pf = precision_score(y_test, y_pred)
roc_auc_pf = roc_auc_score(y_test, log_reg.predict_proba(X_test)[:, 1])  # Extract probabilities for the positive class

print("Precision on test data :", precision_pf)
print("ROC AUC score on test data :", roc_auc_pf)
```

```
Precision on test data : 0.9770114942528736
ROC AUC score on test data : 0.9761108456760631
```


Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.98)

Instead of undersampling and near-miss sampling then logistic regression, we can also choose oversampling with SMOTE then fitting into logistical regression.

```python
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score, roc_auc_score
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Assuming df contains your original DataFrame with 'Class' as the target variable

# Step 1: Split into features and target variable
X = df.drop('Class', axis=1)
y = df['Class']

# Step 2: Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Perform oversampling using SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)

# Step 4: Split oversampled data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)

# Step 5: Train logistic regression model
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

# Step 6: Make predictions on the test data
y_pred = log_reg.predict(X_test)

# Step 7: Calculate precision and ROC-AUC score
precision_pf = precision_score(y_test, y_pred)
roc_auc_pf = roc_auc_score(y_test, log_reg.predict_proba(X_test)[:, 1])  # Extract probabilities for the positive class

print("Precision on test data :", precision_pf)
print("ROC AUC score on test data :", roc_auc_pf)
```

Instead of undersampling and near-miss sampling then logistic regression, we can also choose oversampling with SMOTE then fitting into logistical regression.

```
Precision on test data : 0.9735495039368613
ROC AUC score on test data : 0.9894235687711506
```

PIPELINE

INTRODUCTION

PRACTICAL MOTIVATION & PROBLEM STATEMENT

EDA

EXPLORATORY DATA ANALYSIS & ANALYTIC VISUALIZATION

ALGORITHMIC OPTIMIZATION

DIMENSION REDUCTION, CLASSIFIERS

MACHINE LEARNING

CLASSIFIERS, MODEL EVALUATION, LEARNING CURVES

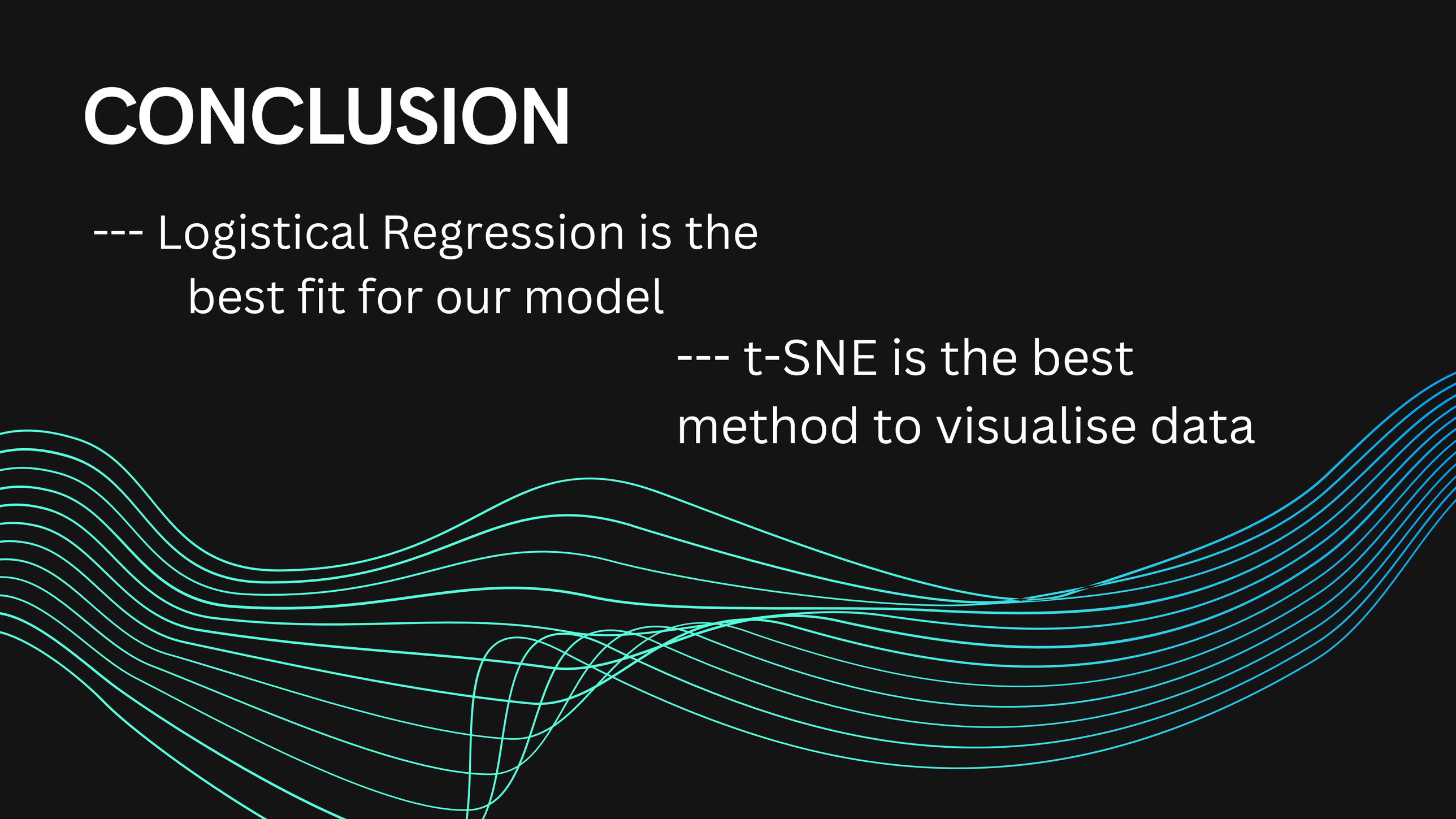
CONCLUSION

STATISTICAL INFERENCE

CONCLUSION

--- Logistical Regression is the best fit for our model

--- t-SNE is the best method to visualise data



METHOD 1:

- UNDERSAMPLING
- LOGISTIC REGRESSION

ROC:.97

METHOD 2:

- OVERSAMPLING WITH SMOTE
- LOGISTIC REGRESSION

ROC:.98

Further Scope

We can remove outliers and visualize while going for the SMOTE method as well.