In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import time
import os

from datetime import datetime

import shap
import lime
from lime import lime_tabular

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, RandomizedSearchCV, GridSearchCV
from sklearn.preprocessing import MinMaxScaler
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn import metrics
from sklearn.metrics import confusion_matrix

from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import precision_recall_curve

from sklearn.cluster import KMeans

import missingno as msno

from fancyimpute import IterativeImputer as MICE
from sklearn.impute import IterativeImputer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import Adam


from sklearn.cluster import DBSCAN
from imblearn.over_sampling import SMOTE
from sklearn.neighbors import NearestNeighbors
from collections import Counter

from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np

from imblearn.over_sampling import KMeansSMOTE
from sklearn.mixture import GaussianMixture


from xgboost import XGBClassifier
from rgf.sklearn import RGFClassifier  # Regularized Greedy Forest
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, roc_auc_score, roc_curve
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

from joblib import dump, load
import logging
```

In [ ]:

```
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

def split_dataset(dataset, target_column, test_size=0.2):
    """
    Split dataset into training and testing sets.
    """
    X = dataset.drop(columns=[target_column])
    y = dataset[target_column]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=42, stratify=y)

    logging.info("Dataset has been split and returned")
    return X_train, X_test, y_train, y_test

def train_ann(X_train, y_train):
    """
    Train an Artificial Neural Network (ANN) on the training data.
    """
    start_time = time.time()
    model = Sequential([
        Input(shape=(X_train.shape[1],)),
        Dense(12, activation='relu'),
        Dense(8, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.fit(X_train, y_train, epochs=150, batch_size=10, verbose=0)
    end_time = time.time()

    logging.info(f"ANN has been trained in {end_time - start_time:.2f} seconds")
    return model

def train_models(X_train, y_train):
    """
    Train multiple models on the training data.
    """
    models = {}
    param_grids = {
        'RandomForest': {
            'n_estimators': [100, 200, 300],
            'max_depth': [None, 10, 20],
            'min_samples_split': [2, 5]
        },
        'XGBoost': {
            'n_estimators': [100, 200, 300],
            'max_depth': [3, 6],
            'learning_rate': [0.01, 0.1]
        },
        'SVM': {
            'C': [0.1, 1, 10],
            'kernel': ['linear', 'rbf']
        },
        'LogisticRegression': {
            'C': [0.1, 1, 10],
            'penalty': ['l2']
        },
        'GradientBoosting': {
            'n_estimators': [100, 200, 300],
            'learning_rate': [0.01, 0.1],
            'max_depth': [3, 5, 7]
        },
        'KNN': {
            'n_neighbors': [3, 5, 7],
            'weights': ['uniform', 'distance']
        }
    }

    models['ANN'] = train_ann(X_train, y_train)

    for model_name, param_grid in param_grids.items():
        start_time = time.time()
        try:
            if model_name == 'RandomForest':
                model = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
            elif model_name == 'XGBoost':
                model = GridSearchCV(XGBClassifier(), param_grid, cv=5)
            elif model_name == 'SVM':
                model = GridSearchCV(SVC(probability=True), param_grid, cv=5)
            elif model_name == 'LogisticRegression':
                model = GridSearchCV(LogisticRegression(), param_grid, cv=5)
            elif model_name == 'GradientBoosting':
                model = GridSearchCV(GradientBoostingClassifier(), param_grid, cv=5)
            elif model_name == 'KNN':
                model = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)

            model.fit(X_train, y_train)
            models[model_name] = model.best_estimator_
            end_time = time.time()
            logging.info(f"{model_name} has been trained in {end_time - start_time:.2f} seconds")
        except Exception as e:
            logging.error(f"Error training {model_name}: {e}")

    try:
        start_time = time.time()
        nb = GaussianNB()
        nb.fit(X_train, y_train)
        models['NaiveBayes'] = nb
        end_time = time.time()
        logging.info(f"Naive Bayes has been trained in {end_time - start_time:.2f} seconds")
    except Exception as e:
        logging.error(f"Error training Naive Bayes: {e}")

    return models

def test_models(models, X_test):
    """
    Test trained models on the test data.
    """
    start_time = time.time()
    predictions = {}
    for name, model in models.items():
        try:
            if name == 'ANN':
                predictions[name] = (model.predict(X_test) > 0.5).astype("int32")
            else:
                predictions[name] = model.predict(X_test)
        except Exception as e:
            logging.error(f"Error testing {name}: {e}")
    end_time = time.time()

    logging.info(f"Models have been tested in {end_time - start_time:.2f} seconds")
    return predictions

def evaluate_models(models, predictions, y_test, X_test):
    """
    Evaluate the performance of models.
    """
    start_time = time.time()
    metrics = {}
    for name, y_pred in predictions.items():
        try:
            accuracy = accuracy_score(y_test, y_pred)
            cm = confusion_matrix(y_test, y_pred)
            f1 = f1_score(y_test, y_pred)
            auc = roc_auc_score(y_test, models[name].predict_proba(X_test)[:, 1]) if name != 'ANN' else roc_auc_score
            metrics[name] = {
                'accuracy': accuracy,
                'confusion_matrix': cm,
                'f1_score': f1,
                'auc_roc': auc
            }
        except Exception as e:
            logging.error(f"Error evaluating {name}: {e}")
    end_time = time.time()

    logging.info(f"Models have been evaluated in {end_time - start_time:.2f} seconds")
    return metrics


def explainability_shap(models, df_name, X_test, feature_names):
    """

    """
    # Ensure X_test is a DataFrame with named columns
    X_test = pd.DataFrame(X_test, columns=feature_names).reset_index(drop=True)

    for name, model in models.items():
        if name == 'ANN':
            continue
        try:
            if name in ['RandomForest', 'XGBoost', 'GradientBoosting']:
                explainer = shap.TreeExplainer(model)

            # No existing methods to analyse other models using SHAP, so only these three models.

            shap_values = explainer.shap_values(X_test)

            plt.figure(figsize=(10, 6))
            shap.summary_plot(shap_values[1] if isinstance(shap_values, list) else shap_values,
                              X_test, plot_type="bar", show=False, max_display=10)
            plt.title(f"Top 10 Most Important Features - {name}")
            plt.tight_layout()
            plt.savefig(f"C:\\Users\\dev\\Desktop\\MSC thesis\\Code\\final_codes\\Lime and shap graphs\\{df_name}_sha
            plt.close()
            logging.info(f"SHAP explanations for {name} created and saved")
        except Exception as e:
            logging.error(f"Error generating SHAP explanations for {name}: {e}")



def explainability_lime(models, df_name, X_train, X_test, feature_names):

    """

    """
    # Ensure X_train and X_test are DataFrames with named columns
    X_train = pd.DataFrame(X_train, columns=feature_names).reset_index(drop=True)
    X_test = pd.DataFrame(X_test, columns=feature_names).reset_index(drop=True)

    explainer = lime.lime_tabular.LimeTabularExplainer(
        X_train.values,  # Use .values to get numpy array
        feature_names=feature_names,
        class_names=['Negative', 'Positive'],
        mode='classification'
    )
    for name, model in models.items():
        if name == 'ANN':
            continue
        try:
            i = np.random.randint(0, X_test.shape[0])
            exp = explainer.explain_instance(
                X_test.iloc[i].values,  # Use .iloc[i].values to get numpy array
                model.predict_proba,
                num_features=6
            )
            feature_importance = pd.DataFrame(exp.as_list(), columns=['Feature', 'Importance'])
            feature_importance['Absolute Importance'] = abs(feature_importance['Importance'])
            feature_importance = feature_importance.sort_values('Absolute Importance', ascending=True)
            plt.figure(figsize=(10, 6))
            colors = ['red' if imp < 0 else 'green' for imp in feature_importance['Importance']]
            plt.barh(feature_importance['Feature'], feature_importance['Importance'], color=colors)
            plt.title(f"LIME Explanation for {name}\nTop 6 Features' Impact on Prediction")
            plt.xlabel('Impact on Prediction (Red = Negative, Green = Positive)')
            plt.tight_layout()
            plt.savefig(f"C:\\Users\\dev\\Desktop\\MSC thesis\\Code\\final_codes\\Lime and shap graphs\\{df_name}_lin
            plt.close()
            logging.info(f"LIME explanation for {name} created and saved")
        except Exception as e:
            logging.error(f"Error generating LIME explanations for {name}: {e}")



def interpret_results(models, X_test, feature_names):
    summary = "Model Interpretation Summary:\n\n"
    for name, model in models.items():
        if name == 'ANN':
            continue
        summary += f"{name} Model:\n"
        summary += f"Feature Importance from {name} Model:\n"
        try:
            if name in ['RandomForest', 'XGBoost', 'GradientBoosting']:
                importances = model.feature_importances_
                importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
                importance_df = importance_df.sort_values('Importance', ascending=False).head(10)
            else:
                importances = model.coef_[0] if hasattr(model, 'coef_') else None
                importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
                importance_df = importance_df.sort_values('Importance', ascending=False).head(10)
            summary += importance_df.to_string(index=False)
            summary += "\n\n"
        except Exception as e:
            logging.error(f"Error interpreting results for {name}: {e}")
    logging.info("Model interpretation summary created")
    return summary


def save_models(models, directory='models'):
    """
    Save trained models to disk.
    """
    if not os.path.exists(directory):
        os.makedirs(directory)
    for name, model in models.items():
        try:
            if name == 'ANN':
                model.save(os.path.join(directory, f'{name}_model.h5'))
            else:
                dump(model, os.path.join(directory, f'{name}_model.joblib'))
            logging.info(f"{name} model saved")
        except Exception as e:
            logging.error(f"Error saving {name} model: {e}")


# Use only if needed to run back with best models
def load_models(directory='models'):
    """
    Load trained models from disk.
    """
    models = {}
    for filename in os.listdir(directory):
        model_name, ext = os.path.splitext(filename)
        try:
            if ext == '.h5':
                models[model_name] = load_model(os.path.join(directory, filename))
            elif ext == '.joblib':
                models[model_name] = load(os.path.join(directory, filename))
            logging.info(f"{model_name} model loaded")
        except Exception as e:
            logging.error(f"Error loading {model_name} model: {e}")
    return models


def main(dataset, target_column, name):
    """
    Main function to train, test, evaluate, and explain models.
    """
    X_train, X_test, y_train, y_test = split_dataset(dataset, target_column)

    # Standardization
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    logging.info("Data has been standardized")

    models = train_models(X_train, y_train)
    predictions = test_models(models, X_test)
    metrics = evaluate_models(models, predictions, y_test, X_test)
    explainability_shap(models, name, X_test, feature_names=dataset.drop(columns=[target_column]).columns)
    explainability_lime(models, name, X_train, X_test, feature_names=dataset.drop(columns=[target_column]).columns)

    save_models(models)
    logging.info("Models have been saved")

    # Interpret results
    summary = interpret_results(models, X_test, feature_names=dataset.drop(columns=[target_column]).columns)
    print(summary)

    return metrics


def modelling_gs(df, name):
    """
    Function to run the main pipeline with the given dataset.
    """
    target_column = 'LABEL'  # Replace with your target column
    results = main(df, target_column, name)
    logging.info("Results have been documented.")
    return results

# To run the modelling function with a dataset 'df':
# results = modelling_gs(df)
```

In [ ]:

```
file_paths = [
    "C:\\Users\\dev\\Desktop\\MSC thesis\\Code\\final_codes\\Processed Datasets\\ADASYN_AE_3_PCA.xlsx",
    "C:\\Users\\dev\\Desktop\\MSC thesis\\Code\\final_codes\\Processed Datasets\\ADASYN_MICE_3_PCA.xlsx",
    "C:\\Users\\dev\\Desktop\\MSC thesis\\Code\\final_codes\\Processed Datasets\\KMSMOTE_AE_3_PCA.xlsx",
    "C:\\Users\\dev\\Desktop\\MSC thesis\\Code\\final_codes\\Processed Datasets\\KMSMOTE_MICE_3_PCA.xlsx",
    "C:\\Users\\dev\\Desktop\\MSC thesis\\Code\\final_codes\\Processed Datasets\\SVMSMOTE_AE_3_PCA.xlsx",
    "C:\\Users\\dev\\Desktop\\MSC thesis\\Code\\final_codes\\Processed Datasets\\SVMSMOTE_MICE_3_PCA.xlsx"
]

# Read the Excel files into dataframes
dfs = [pd.read_excel(file_path) for file_path in file_paths]

print("Datasets are read into dataframes")

tot_start_time = time.time()
start_time = time.time()
# Store results in variables
results_ADASYN_AE_3_PCA = modelling_gs(dfs[0], "ADASYN_AE_3_PCA" )
end_time = time.time()  # End timing
elapsed_time = (end_time - start_time) / 60
print("________________________________________________________________________________")
print(f" Total time taken by ADASYN_AE_3_PCA: {elapsed_time:.2f} mins")

start_time = time.time()
results_ADASYN_MICE_3_PCA = modelling_gs(dfs[1], "ADASYN_MICE_3_PCA")

end_time = time.time()  # End timing
elapsed_time = (end_time - start_time) / 60
print("________________________________________________________________________________")
print(f" Total time taken by ADASYN_MICE_3_PCA: {elapsed_time:.2f} mins")

start_time = time.time()
results_KMSMOTE_AE_3_PCA = modelling_gs(dfs[2], "KMSMOTE_AE_3_PCA")

end_time = time.time()  # End timing
elapsed_time = (end_time - start_time) / 60
print("________________________________________________________________________________")
print(f" Total time taken by KMSMOTE_AE_3_PCA: {elapsed_time:.2f} mins")

start_time = time.time()
results_KMSMOTE_MICE_3_PCA = modelling_gs(dfs[3], "KMSMOTE_MICE_3_PCA")

end_time = time.time()  # End timing
elapsed_time = (end_time - start_time) / 60
print("________________________________________________________________________________")
print(f" Total time taken by KMSMOTE_MICE_3_PCA: {elapsed_time:.2f} mins")

start_time = time.time()
results_SVMSMOTE_AE_3_PCA = modelling_gs(dfs[4], "SVMSMOTE_AE_3_PCA")
```

```python
end_time = time.time()  # End timing
elapsed_time = (end_time - start_time) / 60
print("_______________________________________________________________________________")
print(f" Total time taken by SVMSMOTE_AE_3_PCA: {elapsed_time:.2f} mins")

start_time = time.time()
results_SVMSMOTE_MICE_3_PCA = modelling_gs(dfs[5], "SVMSMOTE_MICE_3_PCA")

end_time = time.time()  # End timing
elapsed_time = (end_time - start_time) / 60
print("_______________________________________________________________________________")
print(f" Total time taken by SVMSMOTE_MICE_3_PCA: {elapsed_time:.2f} mins")


print(" ")
print("_______________________________________________________________________________")
tot_end_time = time.time()  # End timing
tot_elapsed_time = (tot_end_time - tot_start_time) / 60
print(f" Total time taken by all the models : {tot_elapsed_time:.2f} mins")

# Print the results with variable names
print("Results for ADASYN_AE_3_PCA:", results_ADASYN_AE_3_PCA)
print("Results for ADASYN_MICE_3_PCA:", results_ADASYN_MICE_3_PCA)
print("Results for KMSMOTE_AE_3_PCA:", results_KMSMOTE_AE_3_PCA)
print("Results for KMSMOTE_MICE_3_PCA:", results_KMSMOTE_MICE_3_PCA)
print("Results for SVMSMOTE_AE_3_PCA:", results_SVMSMOTE_AE_3_PCA)
print("Results for SVMSMOTE_MICE_3_PCA:", results_SVMSMOTE_MICE_3_PCA)
```

```
2024-07-07 09:28:33,699 - INFO - Dataset has been split and returned
2024-07-07 09:28:33,704 - INFO - Data has been standardized
Datasets are read into dataframes
2024-07-07 09:30:56,632 - INFO - ANN has been trained in 142.93 seconds
2024-07-07 09:43:48,365 - INFO - RandomForest has been trained in 771.73 seconds
2024-07-07 09:44:00,790 - INFO - XGBoost has been trained in 12.42 seconds
2024-07-07 09:54:50,891 - INFO - SVM has been trained in 650.10 seconds
2024-07-07 09:54:51,469 - INFO - LogisticRegression has been trained in 0.58 seconds
2024-07-07 10:32:21,092 - INFO - GradientBoosting has been trained in 2249.62 seconds
2024-07-07 10:32:22,918 - INFO - KNN has been trained in 1.83 seconds
2024-07-07 10:32:22,925 - INFO - Naive Bayes has been trained in 0.01 seconds
126/126 ━━━━━━━━━━━━━━━━━━━━ 0s 741us/step
2024-07-07 10:32:24,648 - INFO - Models have been tested in 1.72 seconds
126/126 ━━━━━━━━━━━━━━━━━━━━ 0s 830us/step
2024-07-07 10:32:26,391 - INFO - Models have been evaluated in 1.74 seconds
2024-07-07 10:33:30,987 - INFO - SHAP explanations for RandomForest created and saved
2024-07-07 10:33:32,109 - INFO - SHAP explanations for XGBoost created and saved
2024-07-07 10:33:33,238 - INFO - SHAP explanations for SVM created and saved
2024-07-07 10:33:34,291 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-07 10:33:49,064 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-07 10:34:04,006 - INFO - SHAP explanations for KNN created and saved
2024-07-07 10:34:18,932 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-07 10:34:19,312 - INFO - LIME explanation for RandomForest created and saved
2024-07-07 10:34:19,613 - INFO - LIME explanation for XGBoost created and saved
2024-07-07 10:34:21,450 - INFO - LIME explanation for SVM created and saved
2024-07-07 10:34:21,729 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-07 10:34:22,081 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-07 10:34:22,447 - INFO - LIME explanation for KNN created and saved
2024-07-07 10:34:22,746 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-07 10:34:22,747 - WARNING - You are saving your model as an HDF5 file via `model.save()` or `keras.saving.sav
e_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `mod
el.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
2024-07-07 10:34:22,773 - INFO - ANN model saved
2024-07-07 10:34:22,807 - INFO - RandomForest model saved
2024-07-07 10:34:22,818 - INFO - XGBoost model saved
2024-07-07 10:34:22,821 - INFO - SVM model saved
2024-07-07 10:34:22,823 - INFO - LogisticRegression model saved
2024-07-07 10:34:22,852 - INFO - GradientBoosting model saved
2024-07-07 10:34:22,855 - INFO - KNN model saved
2024-07-07 10:34:22,857 - INFO - NaiveBayes model saved
2024-07-07 10:34:22,858 - INFO - Models have been saved
2024-07-07 10:34:22,880 - INFO - Model interpretation summary created
2024-07-07 10:34:22,882 - INFO - Results have been documented.
2024-07-07 10:34:22,900 - INFO - Dataset has been split and returned
2024-07-07 10:34:22,910 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.286178
Liquidity_and_Coverage_Ratios_PC1    0.159292
      Cost_and_Expense_Ratios_PC1    0.097693
      Cost_and_Expense_Ratios_PC2    0.091679
         Profitability_Ratios_PC1    0.081257
              Activity_Ratios_PC1    0.052102
              Activity_Ratios_PC2    0.037529
Liquidity_and_Coverage_Ratios_PC2    0.032083
             Cash_Flow_Ratios_PC1    0.025838
             Cash_Flow_Ratios_PC2    0.024492

XGBoost Model:
Feature Importance from XGBoost Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.442036
      Cost_and_Expense_Ratios_PC1    0.099987
         Profitability_Ratios_PC1    0.080053
              Activity_Ratios_PC1    0.048228
Liquidity_and_Coverage_Ratios_PC2    0.043795
Liquidity_and_Coverage_Ratios_PC1    0.039131
             Cash_Flow_Ratios_PC1    0.031891
             Per_Share_Ratios_PC2    0.031738
              Activity_Ratios_PC2    0.030428
         Profitability_Ratios_PC2    0.028905

SVM Model:
Feature Importance from SVM Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

LogisticRegression Model:
Feature Importance from LogisticRegression Model:
                          Feature  Importance
             Per_Share_Ratios_PC1    4.976303
              Leverage_Ratios_PC1    1.882347
         Profitability_Ratios_PC2    1.088192
             Per_Share_Ratios_PC2    0.908395
Liquidity_and_Coverage_Ratios_PC2    0.013621
                Growth_Ratios_PC2   -0.025563
              Leverage_Ratios_PC2   -0.140990
              Activity_Ratios_PC2   -0.291410
             Cash_Flow_Ratios_PC2   -0.545816
      Cost_and_Expense_Ratios_PC2   -0.685676

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.576649
      Cost_and_Expense_Ratios_PC1    0.075725
         Profitability_Ratios_PC1    0.062908
              Activity_Ratios_PC1    0.054575
Liquidity_and_Coverage_Ratios_PC2    0.043115
              Activity_Ratios_PC2    0.034843
Liquidity_and_Coverage_Ratios_PC1    0.034230
             Per_Share_Ratios_PC1    0.021382
             Cash_Flow_Ratios_PC1    0.018732
         Profitability_Ratios_PC2    0.018615

KNN Model:
Feature Importance from KNN Model:
                          Feature Importance
```

```
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

NaiveBayes Model:
Feature Importance from NaiveBayes Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None


_______________________________________________________________________________
 Total time taken by ADASYN_AE_3_PCA: 65.82 mins
```

```
2024-07-07 10:37:29,330 - INFO - ANN has been trained in 186.42 seconds
2024-07-07 10:56:58,129 - INFO - RandomForest has been trained in 1168.80 seconds
2024-07-07 10:57:12,990 - INFO - XGBoost has been trained in 14.86 seconds
2024-07-07 11:20:37,983 - INFO - SVM has been trained in 1404.99 seconds
2024-07-07 11:20:38,685 - INFO - LogisticRegression has been trained in 0.70 seconds
2024-07-07 12:11:53,191 - INFO - GradientBoosting has been trained in 3074.50 seconds
2024-07-07 12:11:56,460 - INFO - KNN has been trained in 3.27 seconds
2024-07-07 12:11:56,471 - INFO - Naive Bayes has been trained in 0.01 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 708us/step
2024-07-07 12:11:59,840 - INFO - Models have been tested in 3.37 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 578us/step
2024-07-07 12:12:03,145 - INFO - Models have been evaluated in 3.30 seconds
2024-07-07 12:13:52,215 - INFO - SHAP explanations for RandomForest created and saved
2024-07-07 12:13:53,758 - INFO - SHAP explanations for XGBoost created and saved
2024-07-07 12:13:55,200 - INFO - SHAP explanations for SVM created and saved
2024-07-07 12:13:56,619 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-07 12:14:17,778 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-07 12:14:38,814 - INFO - SHAP explanations for KNN created and saved
2024-07-07 12:14:59,893 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-07 12:15:00,316 - INFO - LIME explanation for RandomForest created and saved
2024-07-07 12:15:00,619 - INFO - LIME explanation for XGBoost created and saved
2024-07-07 12:15:03,519 - INFO - LIME explanation for SVM created and saved
2024-07-07 12:15:03,846 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-07 12:15:04,190 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-07 12:15:04,617 - INFO - LIME explanation for KNN created and saved
2024-07-07 12:15:04,910 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-07 12:15:04,911 - WARNING - You are saving your model as an HDF5 file via `model.save()` or `keras.saving.sav
e_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `mod
el.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
2024-07-07 12:15:04,941 - INFO - ANN model saved
2024-07-07 12:15:04,988 - INFO - RandomForest model saved
2024-07-07 12:15:04,998 - INFO - XGBoost model saved
2024-07-07 12:15:05,001 - INFO - SVM model saved
2024-07-07 12:15:05,003 - INFO - LogisticRegression model saved
2024-07-07 12:15:05,038 - INFO - GradientBoosting model saved
2024-07-07 12:15:05,042 - INFO - KNN model saved
2024-07-07 12:15:05,043 - INFO - NaiveBayes model saved
2024-07-07 12:15:05,044 - INFO - Models have been saved
2024-07-07 12:15:05,067 - INFO - Model interpretation summary created
2024-07-07 12:15:05,071 - INFO - Results have been documented.
2024-07-07 12:15:05,091 - INFO - Dataset has been split and returned
2024-07-07 12:15:05,101 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.254758
Liquidity_and_Coverage_Ratios_PC1    0.179359
      Cost_and_Expense_Ratios_PC1    0.153682
      Cost_and_Expense_Ratios_PC2    0.071484
Liquidity_and_Coverage_Ratios_PC2    0.055596
         Profitability_Ratios_PC1    0.036778
             Per_Share_Ratios_PC2    0.033949
              Activity_Ratios_PC1    0.030313
                Growth_Ratios_PC1    0.028375
         Profitability_Ratios_PC2    0.026236

XGBoost Model:
Feature Importance from XGBoost Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.401111
      Cost_and_Expense_Ratios_PC1    0.139944
Liquidity_and_Coverage_Ratios_PC1    0.059314
Liquidity_and_Coverage_Ratios_PC2    0.046645
                Growth_Ratios_PC1    0.043735
             Per_Share_Ratios_PC1    0.042803
              Activity_Ratios_PC1    0.039137
             Per_Share_Ratios_PC2    0.033031
             Cash_Flow_Ratios_PC2    0.029865
         Profitability_Ratios_PC2    0.029408

SVM Model:
Feature Importance from SVM Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

LogisticRegression Model:
Feature Importance from LogisticRegression Model:
                          Feature  Importance
             Per_Share_Ratios_PC1    3.245274
         Profitability_Ratios_PC2    3.064337
              Leverage_Ratios_PC1    1.457914
         Profitability_Ratios_PC1    1.093241
Liquidity_and_Coverage_Ratios_PC2    0.165945
      Cost_and_Expense_Ratios_PC2    0.067829
              Leverage_Ratios_PC2   -0.006824
              Activity_Ratios_PC1   -0.067760
                Growth_Ratios_PC2   -0.142598
             Per_Share_Ratios_PC2   -0.181289

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.524390
      Cost_and_Expense_Ratios_PC1    0.139134
Liquidity_and_Coverage_Ratios_PC2    0.054392
Liquidity_and_Coverage_Ratios_PC1    0.052117
         Profitability_Ratios_PC2    0.032017
                Growth_Ratios_PC1    0.029256
              Activity_Ratios_PC1    0.027038
             Cash_Flow_Ratios_PC2    0.026829
         Profitability_Ratios_PC1    0.025694
             Per_Share_Ratios_PC2    0.020287

KNN Model:
Feature Importance from KNN Model:
                          Feature Importance
```

```
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

NaiveBayes Model:
Feature Importance from NaiveBayes Model:
                           Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None


________________________________________________________________________________
 Total time taken by ADASYN_MICE_3_PCA: 100.70 mins
```

```
2024-07-07 12:18:18,427 - INFO - ANN has been trained in 193.33 seconds
2024-07-07 12:37:08,929 - INFO - RandomForest has been trained in 1130.50 seconds
2024-07-07 12:37:22,754 - INFO - XGBoost has been trained in 13.83 seconds
2024-07-07 12:52:21,695 - INFO - SVM has been trained in 898.94 seconds
2024-07-07 12:52:22,370 - INFO - LogisticRegression has been trained in 0.68 seconds
2024-07-07 13:43:06,913 - INFO - GradientBoosting has been trained in 3044.54 seconds
2024-07-07 13:43:10,012 - INFO - KNN has been trained in 3.10 seconds
2024-07-07 13:43:10,020 - INFO - Naive Bayes has been trained in 0.01 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 659us/step
2024-07-07 13:43:12,633 - INFO - Models have been tested in 2.61 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 602us/step
2024-07-07 13:43:15,239 - INFO - Models have been evaluated in 2.60 seconds
2024-07-07 13:47:56,422 - INFO - SHAP explanations for RandomForest created and saved
2024-07-07 13:47:57,905 - INFO - SHAP explanations for XGBoost created and saved
2024-07-07 13:47:59,308 - INFO - SHAP explanations for SVM created and saved
2024-07-07 13:48:00,742 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-07 13:48:22,047 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-07 13:48:43,132 - INFO - SHAP explanations for KNN created and saved
2024-07-07 13:49:04,254 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-07 13:49:04,720 - INFO - LIME explanation for RandomForest created and saved
2024-07-07 13:49:05,120 - INFO - LIME explanation for XGBoost created and saved
2024-07-07 13:49:07,247 - INFO - LIME explanation for SVM created and saved
2024-07-07 13:49:07,543 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-07 13:49:07,884 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-07 13:49:08,324 - INFO - LIME explanation for KNN created and saved
2024-07-07 13:49:08,612 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-07 13:49:08,613 - WARNING - You are saving your model as an HDF5 file via `model.save()` or `keras.saving.sav
e_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `mod
el.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
2024-07-07 13:49:08,634 - INFO - ANN model saved
2024-07-07 13:49:08,722 - INFO - RandomForest model saved
2024-07-07 13:49:08,731 - INFO - XGBoost model saved
2024-07-07 13:49:08,734 - INFO - SVM model saved
2024-07-07 13:49:08,737 - INFO - LogisticRegression model saved
2024-07-07 13:49:08,765 - INFO - GradientBoosting model saved
2024-07-07 13:49:08,769 - INFO - KNN model saved
2024-07-07 13:49:08,770 - INFO - NaiveBayes model saved
2024-07-07 13:49:08,771 - INFO - Models have been saved
2024-07-07 13:49:08,805 - INFO - Model interpretation summary created
2024-07-07 13:49:08,809 - INFO - Results have been documented.
2024-07-07 13:49:08,821 - INFO - Dataset has been split and returned
2024-07-07 13:49:08,828 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.294361
Liquidity_and_Coverage_Ratios_PC1    0.189178
      Cost_and_Expense_Ratios_PC1    0.140855
Liquidity_and_Coverage_Ratios_PC2    0.074524
      Cost_and_Expense_Ratios_PC2    0.070339
         Profitability_Ratios_PC1    0.039690
             Cash_Flow_Ratios_PC2    0.033981
             Cash_Flow_Ratios_PC1    0.026100
              Activity_Ratios_PC2    0.020516
         Profitability_Ratios_PC2    0.020387

XGBoost Model:
Feature Importance from XGBoost Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.611000
      Cost_and_Expense_Ratios_PC1    0.083739
              Activity_Ratios_PC1    0.039036
Liquidity_and_Coverage_Ratios_PC1    0.034878
Liquidity_and_Coverage_Ratios_PC2    0.034268
             Cash_Flow_Ratios_PC2    0.029417
              Activity_Ratios_PC2    0.023815
             Per_Share_Ratios_PC2    0.020951
         Profitability_Ratios_PC2    0.020578
             Cash_Flow_Ratios_PC1    0.019791

SVM Model:
Feature Importance from SVM Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

LogisticRegression Model:
Feature Importance from LogisticRegression Model:
                          Feature  Importance
             Per_Share_Ratios_PC1    6.375538
         Profitability_Ratios_PC2    4.673530
      Cost_and_Expense_Ratios_PC1    3.302927
              Leverage_Ratios_PC1    2.498492
             Per_Share_Ratios_PC2    2.270628
         Profitability_Ratios_PC1    0.825373
             Cash_Flow_Ratios_PC2    0.150386
                Growth_Ratios_PC2    0.003326
              Leverage_Ratios_PC2   -0.133484
Liquidity_and_Coverage_Ratios_PC2   -0.204408

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.678024
      Cost_and_Expense_Ratios_PC1    0.093020
             Cash_Flow_Ratios_PC2    0.040772
Liquidity_and_Coverage_Ratios_PC1    0.031648
Liquidity_and_Coverage_Ratios_PC2    0.028278
              Activity_Ratios_PC1    0.022041
         Profitability_Ratios_PC2    0.016642
             Cash_Flow_Ratios_PC1    0.016279
         Profitability_Ratios_PC1    0.015263
             Per_Share_Ratios_PC2    0.012630

KNN Model:
Feature Importance from KNN Model:
                          Feature Importance
```

```
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

NaiveBayes Model:
Feature Importance from NaiveBayes Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None


________________________________________________________________________________
 Total time taken by KMSMOTE_AE_3_PCA: 94.06 mins
```

```
2024-07-07 13:51:16,803 - INFO - ANN has been trained in 127.97 seconds
2024-07-07 14:04:12,362 - INFO - RandomForest has been trained in 775.56 seconds
2024-07-07 14:04:24,495 - INFO - XGBoost has been trained in 12.13 seconds
2024-07-07 14:12:46,293 - INFO - SVM has been trained in 501.80 seconds
2024-07-07 14:12:46,789 - INFO - LogisticRegression has been trained in 0.50 seconds
2024-07-07 14:49:46,490 - INFO - GradientBoosting has been trained in 2219.70 seconds
2024-07-07 14:49:48,387 - INFO - KNN has been trained in 1.89 seconds
2024-07-07 14:49:48,395 - INFO - Naive Bayes has been trained in 0.01 seconds
126/126 ━━━━━━━━━━━━━━━━━━━━ 0s 902us/step
2024-07-07 14:49:49,943 - INFO - Models have been tested in 1.55 seconds
126/126 ━━━━━━━━━━━━━━━━━━━━ 0s 641us/step
2024-07-07 14:49:51,423 - INFO - Models have been evaluated in 1.48 seconds
2024-07-07 14:51:48,709 - INFO - SHAP explanations for RandomForest created and saved
2024-07-07 14:51:50,454 - INFO - SHAP explanations for XGBoost created and saved
2024-07-07 14:51:51,520 - INFO - SHAP explanations for SVM created and saved
2024-07-07 14:51:52,568 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-07 14:52:06,578 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-07 14:52:20,833 - INFO - SHAP explanations for KNN created and saved
2024-07-07 14:52:35,061 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-07 14:52:35,507 - INFO - LIME explanation for RandomForest created and saved
2024-07-07 14:52:35,818 - INFO - LIME explanation for XGBoost created and saved
2024-07-07 14:52:37,338 - INFO - LIME explanation for SVM created and saved
2024-07-07 14:52:37,632 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-07 14:52:37,976 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-07 14:52:38,375 - INFO - LIME explanation for KNN created and saved
2024-07-07 14:52:38,676 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-07 14:52:38,677 - WARNING - You are saving your model as an HDF5 file via `model.save()` or `keras.saving.sav
e_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `mod
el.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
2024-07-07 14:52:38,695 - INFO - ANN model saved
2024-07-07 14:52:38,754 - INFO - RandomForest model saved
2024-07-07 14:52:38,764 - INFO - XGBoost model saved
2024-07-07 14:52:38,766 - INFO - SVM model saved
2024-07-07 14:52:38,768 - INFO - LogisticRegression model saved
2024-07-07 14:52:38,796 - INFO - GradientBoosting model saved
2024-07-07 14:52:38,800 - INFO - KNN model saved
2024-07-07 14:52:38,801 - INFO - NaiveBayes model saved
2024-07-07 14:52:38,802 - INFO - Models have been saved
2024-07-07 14:52:38,828 - INFO - Model interpretation summary created
2024-07-07 14:52:38,830 - INFO - Results have been documented.
2024-07-07 14:52:38,848 - INFO - Dataset has been split and returned
2024-07-07 14:52:38,861 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.319627
Liquidity_and_Coverage_Ratios_PC1    0.199593
      Cost_and_Expense_Ratios_PC1    0.103421
         Profitability_Ratios_PC1    0.083072
      Cost_and_Expense_Ratios_PC2    0.060213
Liquidity_and_Coverage_Ratios_PC2    0.036117
              Activity_Ratios_PC1    0.028794
              Activity_Ratios_PC2    0.025276
             Cash_Flow_Ratios_PC2    0.024993
         Profitability_Ratios_PC2    0.019408

XGBoost Model:
Feature Importance from XGBoost Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.586943
      Cost_and_Expense_Ratios_PC1    0.078286
              Activity_Ratios_PC1    0.038539
         Profitability_Ratios_PC1    0.037865
             Per_Share_Ratios_PC1    0.033482
              Activity_Ratios_PC2    0.030563
             Cash_Flow_Ratios_PC2    0.029050
Liquidity_and_Coverage_Ratios_PC1    0.028263
                Growth_Ratios_PC1    0.024913
             Per_Share_Ratios_PC2    0.023002

SVM Model:
Feature Importance from SVM Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

LogisticRegression Model:
Feature Importance from LogisticRegression Model:
                          Feature  Importance
             Per_Share_Ratios_PC1    4.372106
              Leverage_Ratios_PC1    2.988116
             Per_Share_Ratios_PC2    0.722223
         Profitability_Ratios_PC2    0.575467
              Activity_Ratios_PC2    0.139349
Liquidity_and_Coverage_Ratios_PC2    0.118345
              Leverage_Ratios_PC2    0.018585
             Cash_Flow_Ratios_PC2   -0.017016
                Growth_Ratios_PC2   -0.017203
      Cost_and_Expense_Ratios_PC2   -0.173062

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.696497
      Cost_and_Expense_Ratios_PC1    0.054372
              Activity_Ratios_PC1    0.036068
         Profitability_Ratios_PC1    0.029556
             Per_Share_Ratios_PC1    0.026434
             Cash_Flow_Ratios_PC2    0.025375
              Activity_Ratios_PC2    0.022513
Liquidity_and_Coverage_Ratios_PC1    0.022277
Liquidity_and_Coverage_Ratios_PC2    0.020215
                Growth_Ratios_PC1    0.017521

KNN Model:
Feature Importance from KNN Model:
                          Feature Importance
```

```
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

NaiveBayes Model:
Feature Importance from NaiveBayes Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None


________________________________________________________________________________
 Total time taken by KMSMOTE_MICE_3_PCA: 63.50 mins
```

```
2024-07-07 14:55:51,360 - INFO - ANN has been trained in 192.50 seconds
2024-07-07 15:13:26,237 - INFO - RandomForest has been trained in 1054.88 seconds
2024-07-07 15:13:40,023 - INFO - XGBoost has been trained in 13.79 seconds
2024-07-07 15:28:05,845 - INFO - SVM has been trained in 865.82 seconds
c:\Users\dev\Desktop\MSC thesis\Code\mscthesis\Lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWa
rning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
2024-07-07 15:28:06,485 - INFO - LogisticRegression has been trained in 0.64 seconds
2024-07-07 16:18:03,256 - INFO - GradientBoosting has been trained in 2996.77 seconds
2024-07-07 16:18:06,263 - INFO - KNN has been trained in 3.01 seconds
2024-07-07 16:18:06,270 - INFO - Naive Bayes has been trained in 0.01 seconds
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 727us/step
```

```
2024-07-07 16:18:08,670 - INFO - Models have been tested in 2.40 seconds
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 578us/step
```

```
2024-07-07 16:18:11,073 - INFO - Models have been evaluated in 2.40 seconds
2024-07-07 16:19:31,035 - INFO - SHAP explanations for RandomForest created and saved
2024-07-07 16:19:32,435 - INFO - SHAP explanations for XGBoost created and saved
2024-07-07 16:19:33,741 - INFO - SHAP explanations for SVM created and saved
2024-07-07 16:19:35,053 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-07 16:19:55,339 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-07 16:20:15,401 - INFO - SHAP explanations for KNN created and saved
2024-07-07 16:20:35,442 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-07 16:20:35,836 - INFO - LIME explanation for RandomForest created and saved
2024-07-07 16:20:36,138 - INFO - LIME explanation for XGBoost created and saved
2024-07-07 16:20:38,169 - INFO - LIME explanation for SVM created and saved
2024-07-07 16:20:38,445 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-07 16:20:38,790 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-07 16:20:39,207 - INFO - LIME explanation for KNN created and saved
2024-07-07 16:20:39,497 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-07 16:20:39,498 - WARNING - You are saving your model as an HDF5 file via `model.save()` or `keras.saving.sav
e_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `mod
el.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
2024-07-07 16:20:39,517 - INFO - ANN model saved
2024-07-07 16:20:39,550 - INFO - RandomForest model saved
2024-07-07 16:20:39,560 - INFO - XGBoost model saved
2024-07-07 16:20:39,563 - INFO - SVM model saved
2024-07-07 16:20:39,565 - INFO - LogisticRegression model saved
2024-07-07 16:20:39,594 - INFO - GradientBoosting model saved
2024-07-07 16:20:39,598 - INFO - KNN model saved
2024-07-07 16:20:39,599 - INFO - NaiveBayes model saved
2024-07-07 16:20:39,600 - INFO - Models have been saved
2024-07-07 16:20:39,620 - INFO - Model interpretation summary created
2024-07-07 16:20:39,623 - INFO - Results have been documented.
2024-07-07 16:20:39,638 - INFO - Dataset has been split and returned
2024-07-07 16:20:39,647 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.247876
Liquidity_and_Coverage_Ratios_PC1    0.231795
      Cost_and_Expense_Ratios_PC1    0.147334
      Cost_and_Expense_Ratios_PC2    0.072290
Liquidity_and_Coverage_Ratios_PC2    0.070489
         Profitability_Ratios_PC1    0.045076
             Cash_Flow_Ratios_PC1    0.034320
             Cash_Flow_Ratios_PC2    0.029709
              Activity_Ratios_PC2    0.019279
              Activity_Ratios_PC1    0.019151

XGBoost Model:
Feature Importance from XGBoost Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.503242
      Cost_and_Expense_Ratios_PC1    0.088409
Liquidity_and_Coverage_Ratios_PC1    0.075224
Liquidity_and_Coverage_Ratios_PC2    0.056867
              Activity_Ratios_PC2    0.036598
             Cash_Flow_Ratios_PC1    0.035002
             Cash_Flow_Ratios_PC2    0.027541
             Per_Share_Ratios_PC1    0.026290
              Activity_Ratios_PC1    0.022082
         Profitability_Ratios_PC2    0.021653

SVM Model:
Feature Importance from SVM Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

LogisticRegression Model:
Feature Importance from LogisticRegression Model:
                 Feature  Importance
     Leverage_Ratios_PC1    1.988271
Profitability_Ratios_PC2    1.659230
    Per_Share_Ratios_PC1    1.142546
    Cash_Flow_Ratios_PC2    0.653570
     Activity_Ratios_PC1    0.160677
Profitability_Ratios_PC1    0.082127
     Leverage_Ratios_PC2    0.045274
    Per_Share_Ratios_PC2   -0.008211
       Growth_Ratios_PC2   -0.022705
     Activity_Ratios_PC2   -0.137229

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.649140
      Cost_and_Expense_Ratios_PC1    0.087957
Liquidity_and_Coverage_Ratios_PC1    0.066636
Liquidity_and_Coverage_Ratios_PC2    0.047281
             Cash_Flow_Ratios_PC1    0.026460
              Activity_Ratios_PC2    0.021061
             Cash_Flow_Ratios_PC2    0.018908
         Profitability_Ratios_PC1    0.016663
             Per_Share_Ratios_PC1    0.011288
                Growth_Ratios_PC1    0.010754

KNN Model:
Feature Importance from KNN Model:
                          Feature Importance
```

```
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

NaiveBayes Model:
Feature Importance from NaiveBayes Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None


________________________________________________________________________________
 Total time taken by SVMSMOTE_AE_3_PCA: 88.01 mins
```

```
2024-07-07 16:22:56,780 - INFO - ANN has been trained in 137.13 seconds
2024-07-07 16:35:05,647 - INFO - RandomForest has been trained in 728.87 seconds
2024-07-07 16:35:18,458 - INFO - XGBoost has been trained in 12.81 seconds
2024-07-07 16:43:04,725 - INFO - SVM has been trained in 466.27 seconds
2024-07-07 16:43:05,260 - INFO - LogisticRegression has been trained in 0.53 seconds
2024-07-07 17:19:03,206 - INFO - GradientBoosting has been trained in 2157.94 seconds
2024-07-07 17:19:04,905 - INFO - KNN has been trained in 1.70 seconds
2024-07-07 17:19:04,910 - INFO - Naive Bayes has been trained in 0.00 seconds
126/126 ━━━━━━━━━━━━━━━━━━━━ 0s 748us/step
2024-07-07 17:19:06,362 - INFO - Models have been tested in 1.45 seconds
126/126 ━━━━━━━━━━━━━━━━━━━━ 0s 652us/step
2024-07-07 17:19:07,861 - INFO - Models have been evaluated in 1.50 seconds
2024-07-07 17:21:37,768 - INFO - SHAP explanations for RandomForest created and saved
2024-07-07 17:21:38,793 - INFO - SHAP explanations for XGBoost created and saved
2024-07-07 17:21:39,840 - INFO - SHAP explanations for SVM created and saved
2024-07-07 17:21:40,824 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-07 17:21:55,030 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-07 17:22:09,190 - INFO - SHAP explanations for KNN created and saved
2024-07-07 17:22:23,489 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-07 17:22:23,941 - INFO - LIME explanation for RandomForest created and saved
2024-07-07 17:22:24,254 - INFO - LIME explanation for XGBoost created and saved
2024-07-07 17:22:25,814 - INFO - LIME explanation for SVM created and saved
2024-07-07 17:22:26,104 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-07 17:22:26,453 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-07 17:22:26,854 - INFO - LIME explanation for KNN created and saved
2024-07-07 17:22:27,141 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-07 17:22:27,142 - WARNING - You are saving your model as an HDF5 file via `model.save()` or `keras.saving.sav
e_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `mod
el.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
2024-07-07 17:22:27,159 - INFO - ANN model saved
2024-07-07 17:22:27,243 - INFO - RandomForest model saved
2024-07-07 17:22:27,252 - INFO - XGBoost model saved
2024-07-07 17:22:27,255 - INFO - SVM model saved
2024-07-07 17:22:27,256 - INFO - LogisticRegression model saved
2024-07-07 17:22:27,282 - INFO - GradientBoosting model saved
2024-07-07 17:22:27,286 - INFO - KNN model saved
2024-07-07 17:22:27,288 - INFO - NaiveBayes model saved
2024-07-07 17:22:27,288 - INFO - Models have been saved
2024-07-07 17:22:27,321 - INFO - Model interpretation summary created
2024-07-07 17:22:27,323 - INFO - Results have been documented.
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.282660
Liquidity_and_Coverage_Ratios_PC1    0.212262
      Cost_and_Expense_Ratios_PC1    0.105820
         Profitability_Ratios_PC1    0.091775
      Cost_and_Expense_Ratios_PC2    0.055777
             Cash_Flow_Ratios_PC1    0.043473
Liquidity_and_Coverage_Ratios_PC2    0.034843
              Activity_Ratios_PC1    0.033217
              Activity_Ratios_PC2    0.025364
             Cash_Flow_Ratios_PC2    0.023190

XGBoost Model:
Feature Importance from XGBoost Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.516219
         Profitability_Ratios_PC1    0.066100
Liquidity_and_Coverage_Ratios_PC1    0.058156
      Cost_and_Expense_Ratios_PC1    0.055363
Liquidity_and_Coverage_Ratios_PC2    0.042555
              Activity_Ratios_PC1    0.040352
             Cash_Flow_Ratios_PC1    0.035763
             Per_Share_Ratios_PC1    0.032786
              Activity_Ratios_PC2    0.027681
         Profitability_Ratios_PC2    0.025099

SVM Model:
Feature Importance from SVM Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

LogisticRegression Model:
Feature Importance from LogisticRegression Model:
                          Feature  Importance
             Per_Share_Ratios_PC1    3.124650
              Leverage_Ratios_PC1    2.961973
      Cost_and_Expense_Ratios_PC2    1.269222
             Per_Share_Ratios_PC2    0.410330
         Profitability_Ratios_PC2    0.367171
Liquidity_and_Coverage_Ratios_PC2    0.209284
              Activity_Ratios_PC2    0.158028
              Leverage_Ratios_PC2    0.022434
                Growth_Ratios_PC2   -0.123096
             Cash_Flow_Ratios_PC2   -0.536345

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.636025
Liquidity_and_Coverage_Ratios_PC1    0.068431
         Profitability_Ratios_PC1    0.052173
              Activity_Ratios_PC1    0.044120
      Cost_and_Expense_Ratios_PC1    0.037730
Liquidity_and_Coverage_Ratios_PC2    0.035531
             Cash_Flow_Ratios_PC1    0.023064
              Activity_Ratios_PC2    0.020921
             Per_Share_Ratios_PC1    0.017063
         Profitability_Ratios_PC2    0.014903

KNN Model:
Feature Importance from KNN Model:
                          Feature Importance
```

```
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None

NaiveBayes Model:
Feature Importance from NaiveBayes Model:
                          Feature Importance
Liquidity_and_Coverage_Ratios_PC1       None
Liquidity_and_Coverage_Ratios_PC2       None
              Leverage_Ratios_PC1       None
              Leverage_Ratios_PC2       None
              Activity_Ratios_PC1       None
              Activity_Ratios_PC2       None
         Profitability_Ratios_PC1       None
         Profitability_Ratios_PC2       None
      Cost_and_Expense_Ratios_PC1       None
      Cost_and_Expense_Ratios_PC2       None


_______________________________________________________________________________
 Total time taken by SVMSMOTE_MICE_3_PCA: 61.79 mins


_______________________________________________________________________________
 Total time taken by all the models : 473.89 mins
Results for ADASYN_AE_3_PCA: {'ANN': {'accuracy': 0.9741293532338309, 'confusion_matrix': array([[1913,  100],
       [   4, 2003]], dtype=int64), 'f1_score': 0.9746958637469586, 'auc_roc': 0.9864663939500373}, 'RandomForest':
{'accuracy': 0.9850746268656716, 'confusion_matrix': array([[1954,   59],
       [   1, 2006]], dtype=int64), 'f1_score': 0.9852652259332023, 'auc_roc': 0.9993600391674345}, 'XGBoost': {'accu
racy': 0.9878109452736319, 'confusion_matrix': array([[1965,   48],
       [   1, 2006]], dtype=int64), 'f1_score': 0.987934006402364, 'auc_roc': 0.9981332103658062}, 'SVM': {'accurac
y': 0.9246268656716418, 'confusion_matrix': array([[1761,  252],
       [  51, 1956]], dtype=int64), 'f1_score': 0.9281138790035587, 'auc_roc': 0.955122421747431}, 'LogisticRegressio
n': {'accuracy': 0.8883084577114427, 'confusion_matrix': array([[1696,  317],
       [ 132, 1875]], dtype=int64), 'f1_score': 0.893069778518695, 'auc_roc': 0.9149986968115322}, 'GradientBoostin
g': {'accuracy': 0.9880597014925373, 'confusion_matrix': array([[1967,   46],
       [   2, 2005]], dtype=int64), 'f1_score': 0.988171513060621, 'auc_roc': 0.9987126527595541}, 'KNN': {'accurac
y': 0.9703980099502487, 'confusion_matrix': array([[1896,  117],
       [   2, 2005]], dtype=int64), 'f1_score': 0.9711794623395495, 'auc_roc': 0.9834469075077764}, 'NaiveBayes': {'a
ccuracy': 0.5378109452736318, 'confusion_matrix': array([[ 213, 1800],
       [  58, 1949]], dtype=int64), 'f1_score': 0.6772063933287005, 'auc_roc': 0.7833683696728613}}
Results for ADASYN_MICE_3_PCA: {'ANN': {'accuracy': 0.9770156877052171, 'confusion_matrix': array([[2618,  124],
       [   2, 2738]], dtype=int64), 'f1_score': 0.9775080328454123, 'auc_roc': 0.992935986306548}, 'RandomForest':
{'accuracy': 0.990879241152864, 'confusion_matrix': array([[2694,   48],
       [   2, 2738]], dtype=int64), 'f1_score': 0.9909518639160333, 'auc_roc': 0.9997954234481731}, 'XGBoost': {'accu
racy': 0.9927033929222912, 'confusion_matrix': array([[2702,   40],
       [   0, 2740]], dtype=int64), 'f1_score': 0.9927536231884058, 'auc_roc': 0.9998087335686563}, 'SVM': {'accurac
y': 0.91353520612915, 'confusion_matrix': array([[2431,  311],
       [ 163, 2577]], dtype=int64), 'f1_score': 0.9157782515991472, 'auc_roc': 0.9508715466892405}, 'LogisticRegressi
on': {'accuracy': 0.850601970083911, 'confusion_matrix': array([[2198,  544],
       [ 275, 2465]], dtype=int64), 'f1_score': 0.857540441815968, 'auc_roc': 0.8971524993744243}, 'GradientBoostin
g': {'accuracy': 0.9905144107989785, 'confusion_matrix': array([[2691,   51],
       [   1, 2739]], dtype=int64), 'f1_score': 0.9905967450271248, 'auc_roc': 0.9997822464288947}, 'KNN': {'accurac
y': 0.9850419554906968, 'confusion_matrix': array([[2660,   82],
       [   0, 2740]], dtype=int64), 'f1_score': 0.9852571017619561, 'auc_roc': 0.9919371283148856}, 'NaiveBayes': {'a
ccuracy': 0.5368478657424298, 'confusion_matrix': array([[ 276, 2466],
       [  73, 2667]], dtype=int64), 'f1_score': 0.6775053981963673, 'auc_roc': 0.7360718240721514}}
Results for KMSMOTE_AE_3_PCA: {'ANN': {'accuracy': 0.9812180889861415, 'confusion_matrix': array([[2643,   99],
       [   4, 2738]], dtype=int64), 'f1_score': 0.9815379100197168, 'auc_roc': 0.9932876809986587}, 'RandomForest':
{'accuracy': 0.9923413566739606, 'confusion_matrix': array([[2703,   39],
       [   3, 2739]], dtype=int64), 'f1_score': 0.9923913043478261, 'auc_roc': 0.9997637846801597}, 'XGBoost': {'accu
racy': 0.9923413566739606, 'confusion_matrix': array([[2703,   39],
       [   3, 2739]], dtype=int64), 'f1_score': 0.9923913043478261, 'auc_roc': 0.9998300207326825}, 'SVM': {'accurac
y': 0.937819110138585, 'confusion_matrix': array([[2524,  218],
       [ 123, 2619]], dtype=int64), 'f1_score': 0.9388779351138197, 'auc_roc': 0.9775250965476918}, 'LogisticRegressi
on': {'accuracy': 0.9088256746900073, 'confusion_matrix': array([[2470,  272],
       [ 228, 2514]], dtype=int64), 'f1_score': 0.9095513748191028, 'auc_roc': 0.9481278605861438}, 'GradientBoostin
g': {'accuracy': 0.9925237053245806, 'confusion_matrix': array([[2704,   38],
       [   3, 2739]], dtype=int64), 'f1_score': 0.9925711179561515, 'auc_roc': 0.9995206531459997}, 'KNN': {'accurac
y': 0.9801239970824216, 'confusion_matrix': array([[2644,   98],
       [  11, 2731]], dtype=int64), 'f1_score': 0.9804343923891582, 'auc_roc': 0.9896042648569595}, 'NaiveBayes': {'a
ccuracy': 0.5574398249452954, 'confusion_matrix': array([[ 388, 2354],
       [  73, 2669]], dtype=int64), 'f1_score': 0.6874436574372182, 'auc_roc': 0.8383312292081307}}
Results for KMSMOTE_MICE_3_PCA: {'ANN': {'accuracy': 0.9709316770186336, 'confusion_matrix': array([[1903,  109],
       [   8, 2005]], dtype=int64), 'f1_score': 0.9716501090380422, 'auc_roc': 0.9898319965946991}, 'RandomForest':
{'accuracy': 0.9836024844720497, 'confusion_matrix': array([[1952,   60],
       [   6, 2007]], dtype=int64), 'f1_score': 0.9838235294117647, 'auc_roc': 0.9992533620927194}, 'XGBoost': {'accu
racy': 0.9878260869565217, 'confusion_matrix': array([[1964,   48],
       [   1, 2012]], dtype=int64), 'f1_score': 0.9879695556101153, 'auc_roc': 0.9981109369614405}, 'SVM': {'accurac
y': 0.9418633540372671, 'confusion_matrix': array([[1825,  187],
       [  47, 1966]], dtype=int64), 'f1_score': 0.943831012962074, 'auc_roc': 0.9731543673873302}, 'LogisticRegressio
n': {'accuracy': 0.9085714285714286, 'confusion_matrix': array([[1778,  234],
       [ 134, 1879]], dtype=int64), 'f1_score': 0.9108095007270964, 'auc_roc': 0.9443532545413067}, 'GradientBoostin
g': {'accuracy': 0.986832298136646, 'confusion_matrix': array([[1965,   47],
       [   6, 2007]], dtype=int64), 'f1_score': 0.986968281288419, 'auc_roc': 0.9985356119616132}, 'KNN': {'accurac
y': 0.9674534161490683, 'confusion_matrix': array([[1882,  130],
       [   1, 2012]], dtype=int64), 'f1_score': 0.9684717208182912, 'auc_roc': 0.9793785720846307}, 'NaiveBayes': {'a
ccuracy': 0.5537888198757764, 'confusion_matrix': array([[ 278, 1734],
       [  62, 1951]], dtype=int64), 'f1_score': 0.6848016848016848, 'auc_roc': 0.8449715270226629}}
Results for SVMSMOTE_AE_3_PCA: {'ANN': {'accuracy': 0.9824945295404814, 'confusion_matrix': array([[2652,   90],
       [   6, 2736]], dtype=int64), 'f1_score': 0.9827586206896551, 'auc_roc': 0.9955201551785686}, 'RandomForest':
{'accuracy': 0.9943471918307805, 'confusion_matrix': array([[2717,   25],
       [   6, 2736]], dtype=int64), 'f1_score': 0.9943667090677812, 'auc_roc': 0.9998259641069758}, 'XGBoost': {'accu
racy': 0.9941648431801605, 'confusion_matrix': array([[2711,   31],
       [   1, 2741]], dtype=int64), 'f1_score': 0.9941965904969169, 'auc_roc': 0.9998797642741353}, 'SVM': {'accurac
y': 0.9367250182348651, 'confusion_matrix': array([[2526,  216],
       [ 131, 2611]], dtype=int64), 'f1_score': 0.9376907882923325, 'auc_roc': 0.9763351086723476}, 'LogisticRegressi
on': {'accuracy': 0.8907731582786287, 'confusion_matrix': array([[2486,  256],
       [ 343, 2399]], dtype=int64), 'f1_score': 0.889012414304243, 'auc_roc': 0.9433863966576598}, 'GradientBoostin
g': {'accuracy': 0.9921590080233407, 'confusion_matrix': array([[2702,   40],
       [   3, 2739]], dtype=int64), 'f1_score': 0.9922115558775584, 'auc_roc': 0.9997527453380725}, 'KNN': {'accurac
y': 0.9886943836615609, 'confusion_matrix': array([[2683,   59],
       [   3, 2739]], dtype=int64), 'f1_score': 0.9888086642599277, 'auc_roc': 0.9945008115911494}, 'NaiveBayes': {'a
ccuracy': 0.5359226841721372, 'confusion_matrix': array([[ 317, 2425],
       [ 120, 2622]], dtype=int64), 'f1_score': 0.6732571575298498, 'auc_roc': 0.7492756595541383}}
Results for SVMSMOTE_MICE_3_PCA: {'ANN': {'accuracy': 0.9749068322981367, 'confusion_matrix': array([[1941,   72],
       [  29, 1983]], dtype=int64), 'f1_score': 0.9751659700024589, 'auc_roc': 0.990200500918977}, 'RandomForest':
{'accuracy': 0.9903105590062112, 'confusion_matrix': array([[1977,   36],
       [   3, 2009]], dtype=int64), 'f1_score': 0.9903869854572344, 'auc_roc': 0.9993890358791119}, 'XGBoost': {'accu
racy': 0.9895652173913043, 'confusion_matrix': array([[1975,   38],
       [   4, 2008]], dtype=int64), 'f1_score': 0.9896500739280434, 'auc_roc': 0.9988526367873237}, 'SVM': {'accurac
y': 0.9411180124223603, 'confusion_matrix': array([[1831,  182],
       [  55, 1957]], dtype=int64), 'f1_score': 0.9429053240183088, 'auc_roc': 0.9738433531942967}, 'LogisticRegressi
on': {'accuracy': 0.906832298136646, 'confusion_matrix': array([[1786,  227],
       [ 148, 1864]], dtype=int64), 'f1_score': 0.9086034608822813, 'auc_roc': 0.9481566882855871}, 'GradientBoostin
g': {'accuracy': 0.991055900621118, 'confusion_matrix': array([[1979,   34],
       [   2, 2010]], dtype=int64), 'f1_score': 0.9911242603550295, 'auc_roc': 0.9990086801594803}, 'KNN': {'accurac
y': 0.9813664596273292, 'confusion_matrix': array([[1940,   73],
       [   2, 2010]], dtype=int64), 'f1_score': 0.9816849816849816, 'auc_roc': 0.9895041079899144}, 'NaiveBayes': {'a
ccuracy': 0.5443478260869565, 'confusion_matrix': array([[ 257, 1756],
       [  78, 1934]], dtype=int64), 'f1_score': 0.6783584707120308, 'auc_roc': 0.8020928082770145}}
<Figure size 1000x600 with 0 Axes>
<Figure size 1000x600 with 0 Axes>
<Figure size 1000x600 with 0 Axes>
<Figure size 1000x600 with 0 Axes>
<Figure size 1000x600 with 0 Axes>
<Figure size 1000x600 with 0 Axes>
```

```
In [ ]:
```

```
In [ ]:
```