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
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, precision_score, re
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':
```

```python
                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)
            precision = precision_score(y_test, y_pred)
            recall = recall_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,
                'precision': precision,
                'recall': recall,
                '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):

    """
    Generate SHAP graphs for each of the models
    - It indicates the contributions of variables for the prediction of each of the models
    - It shows how variabels / features affect the model performance

    """
    # 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):

    """
    Generates LIME graphs for each of the models
    - This shows the influence of features for the model in classifying the instances
    - Unlike SHAP, this also shows the direction / influence of the variables on each of the classes

    """
    # 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}_lir
            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):

    """
    This shows the importance and the influence of the features in predictions of each of the models
    """

    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_RF_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_RF_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_RF_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_RF_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_RF_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")

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_RF_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)
```

```
2024-07-17 14:38:46,181 - INFO - Dataset has been split and returned
2024-07-17 14:38:46,189 - INFO - Data has been standardized
Datasets are read into dataframes
2024-07-17 14:42:05,117 - INFO - ANN has been trained in 198.93 seconds
2024-07-17 15:01:21,385 - INFO - RandomForest has been trained in 1156.27 seconds
2024-07-17 15:01:37,186 - INFO - XGBoost has been trained in 15.80 seconds
2024-07-17 15:21:45,852 - INFO - SVM has been trained in 1208.66 seconds
2024-07-17 15:21:46,519 - INFO - LogisticRegression has been trained in 0.67 seconds
2024-07-17 16:13:41,153 - INFO - GradientBoosting has been trained in 3114.63 seconds
2024-07-17 16:13:44,085 - INFO - KNN has been trained in 2.93 seconds
2024-07-17 16:13:44,095 - INFO - Naive Bayes has been trained in 0.01 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 658us/step
2024-07-17 16:13:47,484 - INFO - Models have been tested in 3.39 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 525us/step
```

```
2024-07-17 16:13:50,806 - INFO - Models have been evaluated in 3.32 seconds
2024-07-17 16:15:33,458 - INFO - SHAP explanations for RandomForest created and saved
2024-07-17 16:15:34,873 - INFO - SHAP explanations for XGBoost created and saved
2024-07-17 16:15:36,217 - INFO - SHAP explanations for SVM created and saved
2024-07-17 16:15:37,559 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-17 16:15:57,601 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-17 16:16:17,580 - INFO - SHAP explanations for KNN created and saved
2024-07-17 16:16:37,611 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-17 16:16:38,018 - INFO - LIME explanation for RandomForest created and saved
2024-07-17 16:16:38,319 - INFO - LIME explanation for XGBoost created and saved
2024-07-17 16:16:41,110 - INFO - LIME explanation for SVM created and saved
2024-07-17 16:16:41,379 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-17 16:16:41,711 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-17 16:16:42,124 - INFO - LIME explanation for KNN created and saved
2024-07-17 16:16:42,406 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-17 16:16:42,407 - 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')`.
2024-07-17 16:16:42,472 - INFO - ANN model saved
2024-07-17 16:16:42,511 - INFO - RandomForest model saved
2024-07-17 16:16:42,523 - INFO - XGBoost model saved
2024-07-17 16:16:42,526 - INFO - SVM model saved
2024-07-17 16:16:42,528 - INFO - LogisticRegression model saved
2024-07-17 16:16:42,558 - INFO - GradientBoosting model saved
2024-07-17 16:16:42,562 - INFO - KNN model saved
2024-07-17 16:16:42,565 - INFO - NaiveBayes model saved
2024-07-17 16:16:42,565 - INFO - Models have been saved
2024-07-17 16:16:42,598 - INFO - Model interpretation summary created
2024-07-17 16:16:42,600 - INFO - Results have been documented.
2024-07-17 16:16:42,619 - INFO - Dataset has been split and returned
2024-07-17 16:16:42,628 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                           Feature  Importance
              Leverage_Ratios_PC1    0.255254
Liquidity_and_Coverage_Ratios_PC1    0.190513
      Cost_and_Expense_Ratios_PC1    0.129747
      Cost_and_Expense_Ratios_PC2    0.094890
Liquidity_and_Coverage_Ratios_PC2    0.056234
         Profitability_Ratios_PC1    0.034530
              Activity_Ratios_PC1    0.030779
             Cash_Flow_Ratios_PC1    0.030629
         Profitability_Ratios_PC2    0.028792
             Cash_Flow_Ratios_PC2    0.026788

XGBoost Model:
Feature Importance from XGBoost Model:
                           Feature  Importance
              Leverage_Ratios_PC1    0.416653
      Cost_and_Expense_Ratios_PC1    0.090932
      Cost_and_Expense_Ratios_PC2    0.081417
Liquidity_and_Coverage_Ratios_PC2    0.051538
Liquidity_and_Coverage_Ratios_PC1    0.051497
              Activity_Ratios_PC1    0.039150
         Profitability_Ratios_PC2    0.031309
                Growth_Ratios_PC2    0.031272
             Cash_Flow_Ratios_PC2    0.030441
             Per_Share_Ratios_PC2    0.030015

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
      Cost_and_Expense_Ratios_PC1    7.740283
         Profitability_Ratios_PC2    4.521865
              Leverage_Ratios_PC1    1.483903
         Profitability_Ratios_PC1    1.313461
             Per_Share_Ratios_PC1    1.018142
             Per_Share_Ratios_PC2    0.350339
Liquidity_and_Coverage_Ratios_PC2    0.049603
                Growth_Ratios_PC2   -0.001767
              Leverage_Ratios_PC2   -0.022443
             Cash_Flow_Ratios_PC2   -0.203653

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                           Feature  Importance
              Leverage_Ratios_PC1    0.523787
      Cost_and_Expense_Ratios_PC1    0.108516
Liquidity_and_Coverage_Ratios_PC2    0.054043
Liquidity_and_Coverage_Ratios_PC1    0.050750
      Cost_and_Expense_Ratios_PC2    0.043904
         Profitability_Ratios_PC2    0.030677
              Activity_Ratios_PC1    0.026468
                Growth_Ratios_PC2    0.023510
             Per_Share_Ratios_PC2    0.022150
             Cash_Flow_Ratios_PC1    0.020952

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: 97.94 mins
2024-07-17 16:19:36,872 - INFO - ANN has been trained in 174.24 seconds
2024-07-17 16:38:27,084 - INFO - RandomForest has been trained in 1130.21 seconds
2024-07-17 16:38:40,761 - INFO - XGBoost has been trained in 13.68 seconds
2024-07-17 16:58:09,063 - INFO - SVM has been trained in 1168.30 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(
2024-07-17 16:58:09,922 - INFO - LogisticRegression has been trained in 0.86 seconds
2024-07-17 17:48:08,553 - INFO - GradientBoosting has been trained in 2998.63 seconds
2024-07-17 17:48:11,497 - INFO - KNN has been trained in 2.94 seconds
2024-07-17 17:48:11,504 - INFO - Naive Bayes has been trained in 0.01 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 636us/step
2024-07-17 17:48:14,518 - INFO - Models have been tested in 3.01 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 531us/step
```

```
2024-07-17 17:48:17,619 - INFO - Models have been evaluated in 3.10 seconds
2024-07-17 17:50:15,900 - INFO - SHAP explanations for RandomForest created and saved
2024-07-17 17:50:17,762 - INFO - SHAP explanations for XGBoost created and saved
2024-07-17 17:50:19,193 - INFO - SHAP explanations for SVM created and saved
2024-07-17 17:50:20,597 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-17 17:50:41,217 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-17 17:51:01,852 - INFO - SHAP explanations for KNN created and saved
2024-07-17 17:51:22,421 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-17 17:51:22,797 - INFO - LIME explanation for RandomForest created and saved
2024-07-17 17:51:23,099 - INFO - LIME explanation for XGBoost created and saved
2024-07-17 17:51:25,745 - INFO - LIME explanation for SVM created and saved
2024-07-17 17:51:26,015 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-17 17:51:26,349 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-17 17:51:26,767 - INFO - LIME explanation for KNN created and saved
2024-07-17 17:51:27,043 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-17 17:51:27,044 - 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')`.
2024-07-17 17:51:27,061 - INFO - ANN model saved
2024-07-17 17:51:27,095 - INFO - RandomForest model saved
2024-07-17 17:51:27,104 - INFO - XGBoost model saved
2024-07-17 17:51:27,108 - INFO - SVM model saved
2024-07-17 17:51:27,110 - INFO - LogisticRegression model saved
2024-07-17 17:51:27,140 - INFO - GradientBoosting model saved
2024-07-17 17:51:27,144 - INFO - KNN model saved
2024-07-17 17:51:27,145 - INFO - NaiveBayes model saved
2024-07-17 17:51:27,146 - INFO - Models have been saved
2024-07-17 17:51:27,166 - INFO - Model interpretation summary created
2024-07-17 17:51:27,168 - INFO - Results have been documented.
2024-07-17 17:51:27,186 - INFO - Dataset has been split and returned
2024-07-17 17:51:27,195 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                          Feature  Importance
      Cost_and_Expense_Ratios_PC1    0.222406
              Leverage_Ratios_PC1    0.193117
Liquidity_and_Coverage_Ratios_PC1    0.136213
      Cost_and_Expense_Ratios_PC2    0.084318
              Leverage_Ratios_PC2    0.059743
Liquidity_and_Coverage_Ratios_PC2    0.054549
         Profitability_Ratios_PC1    0.034414
              Activity_Ratios_PC1    0.029158
              Activity_Ratios_PC2    0.027616
             Cash_Flow_Ratios_PC1    0.026494

XGBoost Model:
Feature Importance from XGBoost Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.434680
      Cost_and_Expense_Ratios_PC1    0.155829
Liquidity_and_Coverage_Ratios_PC2    0.058382
Liquidity_and_Coverage_Ratios_PC1    0.056347
             Cash_Flow_Ratios_PC1    0.035663
              Activity_Ratios_PC1    0.030532
             Per_Share_Ratios_PC2    0.029561
                Growth_Ratios_PC1    0.026086
              Activity_Ratios_PC2    0.024546
              Leverage_Ratios_PC2    0.024095

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   11.271955
         Profitability_Ratios_PC2    3.747069
              Leverage_Ratios_PC1    1.999951
         Profitability_Ratios_PC1    0.542455
Liquidity_and_Coverage_Ratios_PC2    0.348404
             Per_Share_Ratios_PC2    0.161889
                Growth_Ratios_PC2   -0.041488
                Growth_Ratios_PC1   -0.287880
              Leverage_Ratios_PC2   -0.446575
             Cash_Flow_Ratios_PC2   -0.482558

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.486754
      Cost_and_Expense_Ratios_PC1    0.176618
Liquidity_and_Coverage_Ratios_PC2    0.066777
Liquidity_and_Coverage_Ratios_PC1    0.049894
             Cash_Flow_Ratios_PC1    0.029585
              Activity_Ratios_PC1    0.027504
             Per_Share_Ratios_PC2    0.022354
         Profitability_Ratios_PC1    0.019646
         Profitability_Ratios_PC2    0.019033
                Growth_Ratios_PC1    0.018284

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: 94.74 mins
```

```
2024-07-17 17:54:21,671 - INFO - ANN has been trained in 174.47 seconds
2024-07-17 18:12:49,182 - INFO - RandomForest has been trained in 1107.51 seconds
2024-07-17 18:13:02,937 - INFO - XGBoost has been trained in 13.75 seconds
2024-07-17 18:27:18,355 - INFO - SVM has been trained in 855.42 seconds
2024-07-17 18:27:19,049 - INFO - LogisticRegression has been trained in 0.69 seconds
2024-07-17 19:18:01,047 - INFO - GradientBoosting has been trained in 3042.00 seconds
2024-07-17 19:18:03,948 - INFO - KNN has been trained in 2.90 seconds
2024-07-17 19:18:03,958 - INFO - Naive Bayes has been trained in 0.01 seconds
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 661us/step
```

```
2024-07-17 19:18:06,496 - INFO - Models have been tested in 2.54 seconds
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 548us/step
```

```
2024-07-17 19:18:09,141 - INFO - Models have been evaluated in 2.65 seconds
2024-07-17 19:19:50,562 - INFO - SHAP explanations for RandomForest created and saved
2024-07-17 19:19:52,058 - INFO - SHAP explanations for XGBoost created and saved
2024-07-17 19:19:53,443 - INFO - SHAP explanations for SVM created and saved
2024-07-17 19:19:54,987 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-17 19:20:16,827 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-17 19:20:38,694 - INFO - SHAP explanations for KNN created and saved
2024-07-17 19:21:01,746 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-17 19:21:02,160 - INFO - LIME explanation for RandomForest created and saved
2024-07-17 19:21:02,488 - INFO - LIME explanation for XGBoost created and saved
2024-07-17 19:21:04,755 - INFO - LIME explanation for SVM created and saved
2024-07-17 19:21:05,062 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-17 19:21:05,406 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-17 19:21:05,853 - INFO - LIME explanation for KNN created and saved
2024-07-17 19:21:06,149 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-17 19:21:06,150 - 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')`.
2024-07-17 19:21:06,167 - INFO - ANN model saved
2024-07-17 19:21:06,202 - INFO - RandomForest model saved
2024-07-17 19:21:06,212 - INFO - XGBoost model saved
2024-07-17 19:21:06,215 - INFO - SVM model saved
2024-07-17 19:21:06,218 - INFO - LogisticRegression model saved
2024-07-17 19:21:06,280 - INFO - GradientBoosting model saved
2024-07-17 19:21:06,286 - INFO - KNN model saved
2024-07-17 19:21:06,288 - INFO - NaiveBayes model saved
2024-07-17 19:21:06,289 - INFO - Models have been saved
2024-07-17 19:21:06,313 - INFO - Model interpretation summary created
2024-07-17 19:21:06,315 - INFO - Results have been documented.
2024-07-17 19:21:06,330 - INFO - Dataset has been split and returned
2024-07-17 19:21:06,339 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.283118
Liquidity_and_Coverage_Ratios_PC1    0.187846
      Cost_and_Expense_Ratios_PC1    0.161780
      Cost_and_Expense_Ratios_PC2    0.073090
Liquidity_and_Coverage_Ratios_PC2    0.068924
         Profitability_Ratios_PC1    0.033355
             Cash_Flow_Ratios_PC2    0.032262
             Cash_Flow_Ratios_PC1    0.024272
              Activity_Ratios_PC2    0.022533
              Activity_Ratios_PC1    0.022351

XGBoost Model:
Feature Importance from XGBoost Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.578870
      Cost_and_Expense_Ratios_PC1    0.096509
Liquidity_and_Coverage_Ratios_PC1    0.043744
              Activity_Ratios_PC1    0.035837
Liquidity_and_Coverage_Ratios_PC2    0.034951
              Activity_Ratios_PC2    0.032172
             Cash_Flow_Ratios_PC2    0.028192
             Per_Share_Ratios_PC1    0.021860
             Per_Share_Ratios_PC2    0.021369
         Profitability_Ratios_PC2    0.018247

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
      Cost_and_Expense_Ratios_PC1    9.647082
         Profitability_Ratios_PC2    6.297514
              Leverage_Ratios_PC1    2.232682
         Profitability_Ratios_PC1    0.871257
             Per_Share_Ratios_PC1    0.778627
             Cash_Flow_Ratios_PC2    0.339968
             Per_Share_Ratios_PC2    0.109813
              Leverage_Ratios_PC2    0.025600
                Growth_Ratios_PC2   -0.006458
Liquidity_and_Coverage_Ratios_PC2   -0.285448

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.667153
      Cost_and_Expense_Ratios_PC1    0.106375
             Cash_Flow_Ratios_PC2    0.038058
Liquidity_and_Coverage_Ratios_PC1    0.034847
Liquidity_and_Coverage_Ratios_PC2    0.027154
         Profitability_Ratios_PC1    0.016159
              Activity_Ratios_PC1    0.015583
              Activity_Ratios_PC2    0.015049
             Per_Share_Ratios_PC1    0.013549
             Cash_Flow_Ratios_PC1    0.013094

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: 89.65 mins
```

```
2024-07-17 19:24:03,842 - INFO - ANN has been trained in 177.50 seconds
2024-07-17 19:43:05,836 - INFO - RandomForest has been trained in 1141.99 seconds
2024-07-17 19:43:20,112 - INFO - XGBoost has been trained in 14.28 seconds
2024-07-17 19:58:06,883 - INFO - SVM has been trained in 886.77 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(
2024-07-17 19:58:07,694 - INFO - LogisticRegression has been trained in 0.81 seconds
2024-07-17 20:49:13,382 - INFO - GradientBoosting has been trained in 3065.69 seconds
2024-07-17 20:49:16,384 - INFO - KNN has been trained in 3.00 seconds
2024-07-17 20:49:16,394 - INFO - Naive Bayes has been trained in 0.01 seconds
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 651us/step
```

```
2024-07-17 20:49:18,980 - INFO - Models have been tested in 2.59 seconds
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 531us/step
```

```
2024-07-17 20:49:21,496 - INFO - Models have been evaluated in 2.51 seconds
2024-07-17 20:52:26,632 - INFO - SHAP explanations for RandomForest created and saved
2024-07-17 20:52:28,033 - INFO - SHAP explanations for XGBoost created and saved
2024-07-17 20:52:29,367 - INFO - SHAP explanations for SVM created and saved
2024-07-17 20:52:30,722 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-17 20:52:50,901 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-17 20:53:10,969 - INFO - SHAP explanations for KNN created and saved
2024-07-17 20:53:31,052 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-17 20:53:31,486 - INFO - LIME explanation for RandomForest created and saved
2024-07-17 20:53:31,796 - INFO - LIME explanation for XGBoost created and saved
2024-07-17 20:53:33,793 - INFO - LIME explanation for SVM created and saved
2024-07-17 20:53:34,069 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-17 20:53:34,399 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-17 20:53:34,830 - INFO - LIME explanation for KNN created and saved
2024-07-17 20:53:35,109 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-17 20:53:35,110 - 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')`.
2024-07-17 20:53:35,137 - INFO - ANN model saved
2024-07-17 20:53:35,199 - INFO - RandomForest model saved
2024-07-17 20:53:35,209 - INFO - XGBoost model saved
2024-07-17 20:53:35,212 - INFO - SVM model saved
2024-07-17 20:53:35,213 - INFO - LogisticRegression model saved
2024-07-17 20:53:35,243 - INFO - GradientBoosting model saved
2024-07-17 20:53:35,247 - INFO - KNN model saved
2024-07-17 20:53:35,249 - INFO - NaiveBayes model saved
2024-07-17 20:53:35,250 - INFO - Models have been saved
2024-07-17 20:53:35,279 - INFO - Model interpretation summary created
2024-07-17 20:53:35,282 - INFO - Results have been documented.
2024-07-17 20:53:35,302 - INFO - Dataset has been split and returned
2024-07-17 20:53:35,313 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                             Feature  Importance
               Leverage_Ratios_PC1    0.311899
Liquidity_and_Coverage_Ratios_PC1    0.191697
      Cost_and_Expense_Ratios_PC1    0.108300
Liquidity_and_Coverage_Ratios_PC2    0.074541
      Cost_and_Expense_Ratios_PC2    0.072452
               Activity_Ratios_PC2    0.042610
          Profitability_Ratios_PC1    0.035484
              Cash_Flow_Ratios_PC2    0.028783
               Activity_Ratios_PC1    0.027260
              Cash_Flow_Ratios_PC1    0.020179

XGBoost Model:
Feature Importance from XGBoost Model:
                             Feature  Importance
               Leverage_Ratios_PC1    0.623866
      Cost_and_Expense_Ratios_PC1    0.082495
               Activity_Ratios_PC2    0.037848
Liquidity_and_Coverage_Ratios_PC1    0.035664
Liquidity_and_Coverage_Ratios_PC2    0.029252
               Activity_Ratios_PC1    0.026884
              Cash_Flow_Ratios_PC2    0.024805
              Per_Share_Ratios_PC1    0.020525
          Profitability_Ratios_PC2    0.019611
                 Growth_Ratios_PC1    0.017180

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    7.041636
          Profitability_Ratios_PC2    5.097689
               Leverage_Ratios_PC1    2.613960
          Profitability_Ratios_PC1    0.234622
              Cash_Flow_Ratios_PC2    0.171366
                 Growth_Ratios_PC2   -0.019144
               Leverage_Ratios_PC2   -0.075011
Liquidity_and_Coverage_Ratios_PC2   -0.130918
               Activity_Ratios_PC1   -0.171433
                 Growth_Ratios_PC1   -0.376129

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                             Feature  Importance
               Leverage_Ratios_PC1    0.685466
      Cost_and_Expense_Ratios_PC1    0.082306
               Activity_Ratios_PC2    0.038302
              Cash_Flow_Ratios_PC2    0.026810
Liquidity_and_Coverage_Ratios_PC1    0.025315
               Activity_Ratios_PC1    0.023662
Liquidity_and_Coverage_Ratios_PC2    0.019982
              Per_Share_Ratios_PC1    0.016936
          Profitability_Ratios_PC1    0.015401
          Profitability_Ratios_PC2    0.013583

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: 92.48 mins
```

```
2024-07-17 20:56:29,466 - INFO - ANN has been trained in 174.15 seconds
2024-07-17 21:14:08,666 - INFO - RandomForest has been trained in 1059.20 seconds
2024-07-17 21:14:22,401 - INFO - XGBoost has been trained in 13.73 seconds
2024-07-17 21:27:23,802 - INFO - SVM has been trained in 781.40 seconds
2024-07-17 21:27:24,399 - INFO - LogisticRegression has been trained in 0.60 seconds
2024-07-17 22:18:09,556 - INFO - GradientBoosting has been trained in 3045.16 seconds
2024-07-17 22:18:12,347 - INFO - KNN has been trained in 2.79 seconds
2024-07-17 22:18:12,354 - INFO - Naive Bayes has been trained in 0.01 seconds
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 625us/step
```

```
2024-07-17 22:18:14,730 - INFO - Models have been tested in 2.37 seconds
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 504us/step
```

```
2024-07-17 22:18:17,164 - INFO - Models have been evaluated in 2.43 seconds
2024-07-17 22:22:08,809 - INFO - SHAP explanations for RandomForest created and saved
2024-07-17 22:22:10,151 - INFO - SHAP explanations for XGBoost created and saved
2024-07-17 22:22:11,434 - INFO - SHAP explanations for SVM created and saved
2024-07-17 22:22:12,702 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-17 22:22:31,538 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-17 22:22:50,487 - INFO - SHAP explanations for KNN created and saved
2024-07-17 22:23:09,328 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-17 22:23:09,764 - INFO - LIME explanation for RandomForest created and saved
2024-07-17 22:23:10,058 - INFO - LIME explanation for XGBoost created and saved
2024-07-17 22:23:12,083 - INFO - LIME explanation for SVM created and saved
2024-07-17 22:23:12,344 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-17 22:23:12,656 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-17 22:23:13,038 - INFO - LIME explanation for KNN created and saved
2024-07-17 22:23:13,317 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-17 22:23:13,317 - 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')`.
2024-07-17 22:23:13,336 - INFO - ANN model saved
2024-07-17 22:23:13,932 - INFO - RandomForest model saved
2024-07-17 22:23:13,941 - INFO - XGBoost model saved
2024-07-17 22:23:13,944 - INFO - SVM model saved
2024-07-17 22:23:13,945 - INFO - LogisticRegression model saved
2024-07-17 22:23:13,971 - INFO - GradientBoosting model saved
2024-07-17 22:23:13,975 - INFO - KNN model saved
2024-07-17 22:23:13,976 - INFO - NaiveBayes model saved
2024-07-17 22:23:13,976 - INFO - Models have been saved
2024-07-17 22:23:14,006 - INFO - Model interpretation summary created
2024-07-17 22:23:14,029 - INFO - Results have been documented.
2024-07-17 22:23:14,043 - INFO - Dataset has been split and returned
2024-07-17 22:23:14,050 - INFO - Data has been standardized
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.272939
Liquidity_and_Coverage_Ratios_PC1    0.224027
      Cost_and_Expense_Ratios_PC1    0.125460
Liquidity_and_Coverage_Ratios_PC2    0.082071
      Cost_and_Expense_Ratios_PC2    0.061652
         Profitability_Ratios_PC1    0.042210
             Cash_Flow_Ratios_PC2    0.035113
             Cash_Flow_Ratios_PC1    0.031949
              Activity_Ratios_PC2    0.018981
              Activity_Ratios_PC1    0.018617

XGBoost Model:
Feature Importance from XGBoost Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.505651
Liquidity_and_Coverage_Ratios_PC1    0.090656
      Cost_and_Expense_Ratios_PC1    0.069397
Liquidity_and_Coverage_Ratios_PC2    0.061130
             Cash_Flow_Ratios_PC1    0.036352
             Per_Share_Ratios_PC2    0.031088
      Cost_and_Expense_Ratios_PC2    0.028701
              Activity_Ratios_PC1    0.023663
             Cash_Flow_Ratios_PC2    0.023219
              Leverage_Ratios_PC2    0.022138

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
         Profitability_Ratios_PC2    3.759163
              Leverage_Ratios_PC1    2.403141
         Profitability_Ratios_PC1    1.123301
             Cash_Flow_Ratios_PC2    0.914991
             Per_Share_Ratios_PC1    0.634283
             Per_Share_Ratios_PC2    0.218783
                Growth_Ratios_PC2    0.011864
              Leverage_Ratios_PC2   -0.032254
Liquidity_and_Coverage_Ratios_PC2   -0.376269
                Growth_Ratios_PC1   -0.567703

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                          Feature  Importance
              Leverage_Ratios_PC1    0.639528
Liquidity_and_Coverage_Ratios_PC1    0.078509
      Cost_and_Expense_Ratios_PC1    0.064649
Liquidity_and_Coverage_Ratios_PC2    0.058737
             Cash_Flow_Ratios_PC1    0.024711
         Profitability_Ratios_PC1    0.018624
             Cash_Flow_Ratios_PC2    0.018143
      Cost_and_Expense_Ratios_PC2    0.017879
              Leverage_Ratios_PC2    0.013430
         Profitability_Ratios_PC2    0.012216

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: 89.65 mins
2024-07-17 22:26:07,730 - INFO - ANN has been trained in 173.68 seconds
2024-07-17 22:49:07,644 - INFO - RandomForest has been trained in 1379.91 seconds
2024-07-17 22:49:26,555 - INFO - XGBoost has been trained in 18.91 seconds
2024-07-17 23:04:13,163 - INFO - SVM has been trained in 886.61 seconds
2024-07-17 23:04:13,901 - INFO - LogisticRegression has been trained in 0.74 seconds
2024-07-17 23:54:20,328 - INFO - GradientBoosting has been trained in 3006.43 seconds
2024-07-17 23:54:23,304 - INFO - KNN has been trained in 2.98 seconds
2024-07-17 23:54:23,312 - INFO - Naive Bayes has been trained in 0.01 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 637us/step
2024-07-17 23:54:25,693 - INFO - Models have been tested in 2.38 seconds
172/172 ━━━━━━━━━━━━━━━━━━━━ 0s 523us/step
2024-07-17 23:54:28,147 - INFO - Models have been evaluated in 2.45 seconds
2024-07-17 23:55:47,404 - INFO - SHAP explanations for RandomForest created and saved
2024-07-17 23:55:48,814 - INFO - SHAP explanations for XGBoost created and saved
2024-07-17 23:55:50,135 - INFO - SHAP explanations for SVM created and saved
2024-07-17 23:55:51,450 - INFO - SHAP explanations for LogisticRegression created and saved
2024-07-17 23:56:10,604 - INFO - SHAP explanations for GradientBoosting created and saved
2024-07-17 23:56:29,610 - INFO - SHAP explanations for KNN created and saved
2024-07-17 23:56:48,580 - INFO - SHAP explanations for NaiveBayes created and saved
2024-07-17 23:56:48,961 - INFO - LIME explanation for RandomForest created and saved
2024-07-17 23:56:49,264 - INFO - LIME explanation for XGBoost created and saved
2024-07-17 23:56:51,266 - INFO - LIME explanation for SVM created and saved
2024-07-17 23:56:51,549 - INFO - LIME explanation for LogisticRegression created and saved
2024-07-17 23:56:51,867 - INFO - LIME explanation for GradientBoosting created and saved
2024-07-17 23:56:52,277 - INFO - LIME explanation for KNN created and saved
2024-07-17 23:56:52,567 - INFO - LIME explanation for NaiveBayes created and saved
2024-07-17 23:56:52,567 - 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')`.
2024-07-17 23:56:52,584 - INFO - ANN model saved
2024-07-17 23:56:52,617 - INFO - RandomForest model saved
2024-07-17 23:56:52,626 - INFO - XGBoost model saved
2024-07-17 23:56:52,629 - INFO - SVM model saved
2024-07-17 23:56:52,631 - INFO - LogisticRegression model saved
2024-07-17 23:56:52,659 - INFO - GradientBoosting model saved
2024-07-17 23:56:52,663 - INFO - KNN model saved
2024-07-17 23:56:52,664 - INFO - NaiveBayes model saved
2024-07-17 23:56:52,665 - INFO - Models have been saved
2024-07-17 23:56:52,687 - INFO - Model interpretation summary created
2024-07-17 23:56:52,690 - INFO - Results have been documented.
```

```
Model Interpretation Summary:

RandomForest Model:
Feature Importance from RandomForest Model:
                            Feature  Importance
               Leverage_Ratios_PC1    0.289327
Liquidity_and_Coverage_Ratios_PC1    0.211446
      Cost_and_Expense_Ratios_PC1    0.133188
Liquidity_and_Coverage_Ratios_PC2    0.077444
      Cost_and_Expense_Ratios_PC2    0.055209
         Profitability_Ratios_PC1    0.040261
             Cash_Flow_Ratios_PC2    0.035374
             Cash_Flow_Ratios_PC1    0.027641
              Activity_Ratios_PC2    0.022320
              Activity_Ratios_PC1    0.022066

XGBoost Model:
Feature Importance from XGBoost Model:
                            Feature  Importance
               Leverage_Ratios_PC1    0.511008
Liquidity_and_Coverage_Ratios_PC1    0.088647
      Cost_and_Expense_Ratios_PC1    0.070526
Liquidity_and_Coverage_Ratios_PC2    0.058024
             Cash_Flow_Ratios_PC1    0.039331
             Cash_Flow_Ratios_PC2    0.036763
             Per_Share_Ratios_PC1    0.034040
              Activity_Ratios_PC1    0.029276
         Profitability_Ratios_PC1    0.024291
         Profitability_Ratios_PC2    0.019675

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    8.920448
         Profitability_Ratios_PC2    4.447893
              Leverage_Ratios_PC1    2.230180
             Per_Share_Ratios_PC2    1.692527
             Cash_Flow_Ratios_PC2    0.357021
         Profitability_Ratios_PC1    0.270891
              Leverage_Ratios_PC2    0.039086
                Growth_Ratios_PC2    0.011821
Liquidity_and_Coverage_Ratios_PC2   -0.338323
      Cost_and_Expense_Ratios_PC2   -0.744421

GradientBoosting Model:
Feature Importance from GradientBoosting Model:
                            Feature  Importance
              Leverage_Ratios_PC1    0.639539
Liquidity_and_Coverage_Ratios_PC1    0.075941
Liquidity_and_Coverage_Ratios_PC2    0.059454
      Cost_and_Expense_Ratios_PC1    0.057762
             Cash_Flow_Ratios_PC1    0.027026
             Cash_Flow_Ratios_PC2    0.026216
         Profitability_Ratios_PC1    0.021057
              Activity_Ratios_PC1    0.018894
             Per_Share_Ratios_PC1    0.018352
         Profitability_Ratios_PC2    0.014034

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: 93.64 mins


______________________________________________________________________________
 Total time taken by all the models : 558.11 mins
Results for ADASYN_AE_3_PCA: {'ANN': {'accuracy': 0.9808464064210143, 'confusion_matrix': array([[2655,   87],
       [  18, 2722]], dtype=int64), 'f1_score': 0.98107767165255, 'precision': 0.9690281238875045, 'recall': 0.993430
6569343065, 'auc_roc': 0.9893288771049956}, 'RandomForest': {'accuracy': 0.9916089018606348, 'confusion_matrix': arra
y([[2697,   45],
       [   1, 2739]], dtype=int64), 'f1_score': 0.9916727009413469, 'precision': 0.9838362068965517, 'recall': 0.9996
350364963503, 'auc_roc': 0.9997610833373265}, 'XGBoost': {'accuracy': 0.9928858080992339, 'confusion_matrix': array
([[2703,   39],
       [   0, 2740]], dtype=int64), 'f1_score': 0.9929335024460954, 'precision': 0.9859661748830515, 'recall': 1.0,
'auc_roc': 0.9997399202457581}, 'SVM': {'accuracy': 0.9126231302444363, 'confusion_matrix': array([[2427,  315],
       [ 164, 2576]], dtype=int64), 'f1_score': 0.9149351802521755, 'precision': 0.8910411622276029, 'recall': 0.9401
459854014599, 'auc_roc': 0.9513972964483274}, 'LogisticRegression': {'accuracy': 0.8657424297701569, 'confusion_matri
x': array([[2245,  497],
       [ 239, 2501]], dtype=int64), 'f1_score': 0.8717323109097247, 'precision': 0.8342228152101401, 'recall': 0.9127
737226277373, 'auc_roc': 0.9053999158800384}, 'GradientBoosting': {'accuracy': 0.9925209777453484, 'confusion_matri
x': array([[2701,   41],
       [   0, 2740]], dtype=int64), 'f1_score': 0.9925738090925557, 'precision': 0.9852571017619561, 'recall': 1.0,
'auc_roc': 0.9996235897927348}, 'KNN': {'accuracy': 0.9844947099598687, 'confusion_matrix': array([[2657,   85],
       [   0, 2740]], dtype=int64), 'f1_score': 0.9847259658580413, 'precision': 0.9699115044247788, 'recall': 1.0,
'auc_roc': 0.9908516879894796}, 'NaiveBayes': {'accuracy': 0.5361182050346589, 'confusion_matrix': array([[ 300, 244
2],
       [ 101, 2639]], dtype=int64), 'f1_score': 0.6748497634573584, 'precision': 0.5193859476481008, 'recall': 0.9631
386861313869, 'auc_roc': 0.7088907611791702}}
Results for ADASYN_MICE_3_PCA: {'ANN': {'accuracy': 0.9808254200146092, 'confusion_matrix': array([[2641,  101],
       [   4, 2730]], dtype=int64), 'f1_score': 0.9811320754716981, 'precision': 0.9643235605793006, 'recall': 0.9985
369422092173, 'auc_roc': 0.9899303393472371}, 'RandomForest': {'accuracy': 0.9919649379108838, 'confusion_matrix': ar
ray([[2698,   44],
       [   0, 2734]], dtype=int64), 'f1_score': 0.9920174165457184, 'precision': 0.9841612670986322, 'recall': 1.0,
'auc_roc': 0.9998263219143326}, 'XGBoost': {'accuracy': 0.9926953981008035, 'confusion_matrix': array([[2702,   40],
       [   0, 2734]], dtype=int64), 'f1_score': 0.9927378358750908, 'precision': 0.9855803893294881, 'recall': 1.0,
'auc_roc': 0.9995956848865917}, 'SVM': {'accuracy': 0.9212929145361578, 'confusion_matrix': array([[2435,  307],
       [ 124, 2610]], dtype=int64), 'f1_score': 0.9237303132188993, 'precision': 0.8947548851559822, 'recall': 0.9546
452084857352, 'auc_roc': 0.9569946648012947}, 'LogisticRegression': {'accuracy': 0.8745434623813002, 'confusion_matri
x': array([[2260,  482],
       [ 205, 2529]], dtype=int64), 'f1_score': 0.8804177545691906, 'precision': 0.8399202922617071, 'recall': 0.9250
182882223847, 'auc_roc': 0.9135752767777725}, 'GradientBoosting': {'accuracy': 0.9921475529583638, 'confusion_matri
x': array([[2699,   43],
       [   0, 2734]], dtype=int64), 'f1_score': 0.9921974233351479, 'precision': 0.9845156643860281, 'recall': 1.0,
'auc_roc': 0.9997149385030176}, 'KNN': {'accuracy': 0.9848429510591673, 'confusion_matrix': array([[2659,   83],
       [   0, 2734]], dtype=int64), 'f1_score': 0.9850477391460998, 'precision': 0.9705360312389066, 'recall': 1.0,
'auc_roc': 0.9916086272388066}, 'NaiveBayes': {'accuracy': 0.5787070854638422, 'confusion_matrix': array([[ 535, 220
7],
       [ 100, 2634]], dtype=int64), 'f1_score': 0.6954455445544554, 'precision': 0.5441024581697996, 'recall': 0.9634
235552304315, 'auc_roc': 0.7325007990259087}}
Results for KMSMOTE_AE_3_PCA: {'ANN': {'accuracy': 0.9697301239970825, 'confusion_matrix': array([[2674,   68],
```

```
       [  98, 2644]], dtype=int64), 'f1_score': 0.9695636230289696, 'precision': 0.9749262536873157, 'recall': 0.9642
596644784829, 'auc_roc': 0.9935529577190538}, 'RandomForest': {'accuracy': 0.9925237053245806, 'confusion_matrix': ar
ray([[2704,   38],
       [   3, 2739]], dtype=int64), 'f1_score': 0.9925711179561515, 'precision': 0.9863161685271876, 'recall': 0.9989
059080962801, 'auc_roc': 0.9997837352983894}, 'XGBoost': {'accuracy': 0.9912472647702407, 'confusion_matrix': array
([[2697,   45],
       [   3, 2739]], dtype=int64), 'f1_score': 0.991313789359392, 'precision': 0.9838362068965517, 'recall': 0.99890
59080962801, 'auc_roc': 0.9998794982658923}, 'SVM': {'accuracy': 0.937454412837345, 'confusion_matrix': array([[2534,
208],
       [ 135, 2607]], dtype=int64), 'f1_score': 0.9382760482274609, 'precision': 0.9261101243339254, 'recall': 0.9507
658643326039, 'auc_roc': 0.9771215620429646}, 'LogisticRegression': {'accuracy': 0.9099197665937272, 'confusion_matri
x': array([[2463,  279],
       [ 215, 2527]], dtype=int64), 'f1_score': 0.910958904109589, 'precision': 0.9005702066999287, 'recall': 0.92159
00802334063, 'auc_roc': 0.9476529693702149}, 'GradientBoosting': {'accuracy': 0.9912472647702407, 'confusion_matrix':
array([[2696,   46],
       [   2, 2740]], dtype=int64), 'f1_score': 0.9913169319826338, 'precision': 0.9834888729361091, 'recall': 0.9992
706053975201, 'auc_roc': 0.9995431308425385}, 'KNN': {'accuracy': 0.9819474835886215, 'confusion_matrix': array([[264
8,   94],
       [   5, 2737]], dtype=int64), 'f1_score': 0.982235779651893, 'precision': 0.9667961850936065, 'recall': 0.99817
65134938001, 'auc_roc': 0.9901743870239051}, 'NaiveBayes': {'accuracy': 0.5565280816921955, 'confusion_matrix': array
([[ 387, 2355],
       [  77, 2665]], dtype=int64), 'f1_score': 0.6866786910590054, 'precision': 0.5308764940239044, 'recall': 0.9719
183078045223, 'auc_roc': 0.8175929605706621}}
Results for KMSMOTE_MICE_3_PCA: {'ANN': {'accuracy': 0.9812180889861415, 'confusion_matrix': array([[2659,   83],
       [  20, 2722]], dtype=int64), 'f1_score': 0.9814314043627186, 'precision': 0.9704099821746881, 'recall': 0.9927
060539752006, 'auc_roc': 0.9937081735288814}, 'RandomForest': {'accuracy': 0.9939824945295405, 'confusion_matrix': ar
ray([[2712,   30],
       [   3, 2739]], dtype=int64), 'f1_score': 0.9940119760479041, 'precision': 0.9891657638136512, 'recall': 0.9989
059080962801, 'auc_roc': 0.9997444325804767}, 'XGBoost': {'accuracy': 0.9934354485776805, 'confusion_matrix': array
([[2708,   34],
       [   2, 2740]], dtype=int64), 'f1_score': 0.9934735315445975, 'precision': 0.9877433309300648, 'recall': 0.9992
706053975201, 'auc_roc': 0.9996499331521286}, 'SVM': {'accuracy': 0.9412837345003647, 'confusion_matrix': array([[253
3,  209],
       [ 113, 2629]], dtype=int64), 'f1_score': 0.9422939068100359, 'precision': 0.9263565891472868, 'recall': 0.9587
892049598833, 'auc_roc': 0.9775431851082202}, 'LogisticRegression': {'accuracy': 0.9144784828592268, 'confusion_matri
x': array([[2475,  267],
       [ 202, 2540]], dtype=int64), 'f1_score': 0.9154802667147234, 'precision': 0.9048806555040969, 'recall': 0.9263
311451495259, 'auc_roc': 0.9521755750167187}, 'GradientBoosting': {'accuracy': 0.9930707512764405, 'confusion_matri
x': array([[2709,   33],
       [   5, 2737]], dtype=int64), 'f1_score': 0.9931059506531205, 'precision': 0.988086642599278, 'recall': 0.99817
65134938001, 'auc_roc': 0.9996637655807677}, 'KNN': {'accuracy': 0.9806710430342815, 'confusion_matrix': array([[264
4,   98],
       [   8, 2734]], dtype=int64), 'f1_score': 0.9809831359885182, 'precision': 0.9653954802259888, 'recall': 0.9970
824215900802, 'auc_roc': 0.9903110487587788}, 'NaiveBayes': {'accuracy': 0.5906272793581328, 'confusion_matrix': arra
y([[ 576, 2166],
       [  79, 2663]], dtype=int64), 'f1_score': 0.7034737815348039, 'precision': 0.551459929592048, 'recall': 0.97118
89132020423, 'auc_roc': 0.8436326404882635}}
Results for SVMSMOTE_AE_3_PCA: {'ANN': {'accuracy': 0.9801239970824216, 'confusion_matrix': array([[2646,   96],
       [  13, 2729]], dtype=int64), 'f1_score': 0.9804203341117298, 'precision': 0.9660176991150442, 'recall': 0.9952
589350838804, 'auc_roc': 0.9950142740023228}, 'RandomForest': {'accuracy': 0.9945295404814004, 'confusion_matrix': ar
ray([[2714,   28],
       [   2, 2740]], dtype=int64), 'f1_score': 0.9945553539019963, 'precision': 0.9898843930635838, 'recall': 0.9992
706053975201, 'auc_roc': 0.9998328138192347}, 'XGBoost': {'accuracy': 0.9950765864332604, 'confusion_matrix': array
([[2717,   25],
       [   2, 2740]], dtype=int64), 'f1_score': 0.9950971490829853, 'precision': 0.9909584086799277, 'recall': 0.9992
706053975201, 'auc_roc': 0.9999237886383624}, 'SVM': {'accuracy': 0.9385485047410649, 'confusion_matrix': array([[253
2,  210],
       [ 127, 2615]], dtype=int64), 'f1_score': 0.9394647027124124, 'precision': 0.9256637168141593, 'recall': 0.9536
834427425237, 'auc_roc': 0.9774444960500436}, 'LogisticRegression': {'accuracy': 0.9042669584245077, 'confusion_matri
x': array([[2500,  242],
       [ 283, 2459]], dtype=int64), 'f1_score': 0.9035458386918979, 'precision': 0.9104035542391706, 'recall': 0.8967
906637490882, 'auc_roc': 0.9536464010946771}, 'GradientBoosting': {'accuracy': 0.9925237053245806, 'confusion_matri
x': array([[2703,   39],
       [   2, 2740]], dtype=int64), 'f1_score': 0.9925738090925557, 'precision': 0.9859661748830515, 'recall': 0.9992
706053975201, 'auc_roc': 0.9996955535658139}, 'KNN': {'accuracy': 0.9899708242159008, 'confusion_matrix': array([[268
8,   54],
       [   1, 2741]], dtype=int64), 'f1_score': 0.9900668231894528, 'precision': 0.9806797853309481, 'recall': 0.9996
3530269876, 'auc_roc': 0.9936007727007443}, 'NaiveBayes': {'accuracy': 0.5340991976659373, 'confusion_matrix': array
([[ 312, 2430],
       [ 125, 2617]], dtype=int64), 'f1_score': 0.6719732956733856, 'precision': 0.5185258569447196, 'recall': 0.9544
128373450036, 'auc_roc': 0.7611977499958769}}
Results for SVMSMOTE_MICE_3_PCA: {'ANN': {'accuracy': 0.9839533187454412, 'confusion_matrix': array([[2666,   76],
       [  12, 2730]], dtype=int64), 'f1_score': 0.9841384282624369, 'precision': 0.9729151817533856, 'recall': 0.9956
236323851203, 'auc_roc': 0.994998845524225}, 'RandomForest': {'accuracy': 0.9941648431801605, 'confusion_matrix': arr
ay([[2713,   29],
```

```
       [   3, 2739]], dtype=int64), 'f1_score': 0.9941923774954627, 'precision': 0.9895231213872833, 'recall': 0.9989
059080962801, 'auc_roc': 0.9999251851816384}, 'XGBoost': {'accuracy': 0.9941648431801605, 'confusion_matrix': array
([[2710,   32],
       [   0, 2742]], dtype=int64), 'f1_score': 0.994198694706309, 'precision': 0.9884643114635905, 'recall': 1.0, 'a
uc_roc': 0.9998125971927618}, 'SVM': {'accuracy': 0.9423778264040846, 'confusion_matrix': array([[2537,  205],
       [ 111, 2631]], dtype=int64), 'f1_score': 0.9433488705629258, 'precision': 0.9277150916784203, 'recall': 0.9595
185995623632, 'auc_roc': 0.9785656542924952}, 'LogisticRegression': {'accuracy': 0.9126549963530269, 'confusion_matri
x': array([[2483,  259],
       [ 220, 2522]], dtype=int64), 'f1_score': 0.9132717725873619, 'precision': 0.9068680330816253, 'recall': 0.9197
665937272064, 'auc_roc': 0.9560932114164353}, 'GradientBoosting': {'accuracy': 0.9934354485776805, 'confusion_matri
x': array([[2706,   36],
       [   0, 2742]], dtype=int64), 'f1_score': 0.9934782608695653, 'precision': 0.9870410367170627, 'recall': 1.0,
'auc_roc': 0.9997488882185481}, 'KNN': {'accuracy': 0.9901531728665208, 'confusion_matrix': array([[2690,   52],
       [   2, 2740]], dtype=int64), 'f1_score': 0.9902421395012649, 'precision': 0.9813753581661891, 'recall': 0.9992
706053975201, 'auc_roc': 0.9952252185390721}, 'NaiveBayes': {'accuracy': 0.5632749817651349, 'confusion_matrix': arra
y([[ 461, 2281],
       [ 114, 2628]], dtype=int64), 'f1_score': 0.6869690236570383, 'precision': 0.5353432470971685, 'recall': 0.9584
245076586433, 'auc_roc': 0.7830454060110416}}
<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 [ ]: