Comparison of Dimensionality Reduction Techniques on Machine Learning

By: Syed Bilal Rizwan

1. Aim

The aim of this project is to compare different dimensionality reduction techniques and their effect on Machine Learning performance. The techniques are tried on 15 datasets to see the general behavior.

2. Datasets Chosen

2.1 Classification Datasets Description:

1. **Marketing Campaign Dataset**: A response model can provide a significant boost to the efficiency of a marketing campaign by increasing responses or reducing expenses. The objective is to predict who will respond to an offer for a product or service.
2. **Credit Card Fraud Dataset**: The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.
3. **Heart Disease Prediction Dataset**: According to the CDC, heart disease is one of the leading causes of death for people of most races in the US (African Americans, American Indians and Alaska Natives, and white people). Originally, the dataset come from the CDC and is a major part of the Behavioral Risk Factor Surveillance System (BRFSS), which conducts annual telephone surveys to gather data on the health status of U.S. residents. It consists of 401,958 rows and 279 columns which are reduced to 20 columns.
4. **Diabetes Dataset**: The dataset represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. It includes over 50 features representing patient and hospital outcomes.
5. **High Income Prediction**: Extraction was done by Barry Becker from the 1994 Census database. A set of clean records was extracted. Prediction task is to determine whether a person makes over 50K a year.
6. **Dry Beans Dataset**: Images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. A total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains. Prediction task is to find out the type of bean it is.
7. **Banknote Authentication Dataset**: Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels.
8. **Audit Data**: The goal of the research is to help the auditors by building a classification model that can predict the fraudulent firm on the basis the present and historical risk factors.

2.2 Regression Dataset Description:

1. **Combined Cycle Power Plant Dataset**: The dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V) to predict the net hourly electrical energy output (EP) of the plant. Prediction task is to predict the Electrical Energy Output.
2. **Energy Efficiency Dataset**: Perform energy analysis using 12 different building shapes simulated in Ecotect. The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters.
3. **QSAR Aquatic Toxicity Dataset**: This dataset was used to develop quantitative regression QSAR models to predict acute aquatic toxicity towards the fish Pimephales promelas (fathead minnow) on a set of 908 chemicals. to predict acute aquatic toxicity towards Daphnia Magna. LC50 data, which is the concentration that causes death in 50% of test D. magna over a test duration of 48 hours, was used as model response.
4. **Bike Sharing Dataset**: Bike sharing systems are new generation of traditional bike rentals where entire process from membership, rental and return has become automatic. Through these systems, user can easily rent a bike from a particular position and return at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousand bicycles. Today, there exists great interest in these systems due to their key role in traffic, environmental and health issues. Goal is to predict the number of bikes given the other variables.
5. **Wine Quality Dataset**: Goal is to predict the quality of wine given the other variables.

6. **Student Performance Dataset**: This data approach student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school related features) and it was collected by using school reports and questionnaires.
7. **Buzz in social media(Tom's Hardware Dataset)**: This dataset contains examples of buzz events from two different social networks: Twitter, and Tom's Hardware, a forum network focusing on modern technology with more conservative dynamics.

3. Background

Training a machine learning model on large datasets require a lot of computational resources and is excessively time consuming as well. To achieve the end-goal in a realistic timeframe, it is important to think of ways to pre-process dataset in a way which leads to less computation and allow scalability. This is where dimensionality reduction techniques come into the picture.

Dimensionality reduction maps a high dimensional dataset into a lower dimensional space without losing information in the dataset. These techniques are used as a pre-processing step before using the dataset for training. There are several renowned DR techniques of which a few are chosen for the comparison analysis below.

3.1 Principal Component Analysis (PCA):

3 different variants of PCA are tried below.

3.1.1 Normal PCA:

PCA converts high dimensional dataset into a lower dimensional dataset while still capturing maximum information in the dataset. It does that by converting n correlated features into k uncorrelated features(components) where k is smaller than n. Furthermore, it ensures that maximum variance is captured by the first component and the second highest variance is captured by the second principal component so this way most of the variance is captured by the first few independent components which are then used to transform the dataset into a lower dimension.

3.1.2 Sparse PCA(spca):

Sparse PCA is another variant of PCA that extract sparse components which can help in reconstruction of data. It overcomes the disadvantage of normal PCA which uses all input features to generate the transformed data, but sparse PCA only uses a few input features to transform the data.

3.1.3 Incremental PCA(ipca):

This is like normal PCA however incremental PCA is for large datasets which might be too large to fit to the memory. Incremental PCA makes a low rank approximation which is not based on the number of samples but only on the number of features.

3.2 Linear Discriminant Analysis (LDA):

Linear Discriminant Analysis utilizes class labels along with the dataset to reduce dimensionality making it a supervised dimensionality reduction technique as opposed to PCA. It is a technique that is used to find linear combination of features that ensure separability of classes. Furthermore, the number of components found are always less than number of classes which means it is a strong dimensionality reduction technique. For example, if LDA was applied on a binary classification dataset, then the resulting components would just be 1. Lastly, this technique is only applicable on classification datasets.

3.3 Singular value Decomposition(SVD):

This technique is like PCA where the only difference is that the matrix factorization is performed on data matrix rather than the covariance matrix which is the case for PCA.

4. Methodology

Analysis is done on 15 datasets consisting of 8 classification and 7 regression datasets. The project code structure is divided into these two parts Classification and Regression. Each part is also divided further into more sections. In the first section, the datasets are loaded. In the second section, the datasets are pre-processed and lastly in the last section, each dataset goes through the Machine learning analysis using Dimensionality Reduction Techniques.

4.1 Pre-Processing

Once datasets are loaded, each dataset is pre-processed according to its need. A bird's eye view of original dataset is printed before starting its pre-processing.

1. Firstly, the datatypes and missing values in each column is checked. If any missing values exist, they are addressed by either removing them or imputing them.
2. Unnecessary columns are removed
3. Categorical variables are one-hot encoded
4. All numerical columns are scaled using a Min-Max Scaler
5. If any additional pre-processing is required by any dataset, it is done in the last step.
6. Predictors are separated from target variable. Convention is to name predictors dataframe as dataset_df and target variable as dataset_classes

Finally, the predictors dataset is printed to see how it looks. This process is repeated for all datasets belonging to that section i.e., classification or regression.

4.2 Machine Learning with Dimensionality Reduction

After all datasets belonging to that section are pre-processed, machine learning is carried out by trying out various dimensionality reduction techniques.

1. Firstly, a pipeline is developed to do the whole analysis and give us final results dataframe. Then, the pipeline is run on all the datasets of that part. Lazy predict library is used to automate running different machine learning models for the classification task and regression task. The pipeline can be broken into 6 parts:

```
i)  Lazy Predict on dataset with original features
ii) Applying PCA and then running Lazy Predict on resulting dataset.
iii) Applying other PCA variants and then running Lazy Predict on resulting dataset.
iv) Applying LDA and then running resulting dataset (only applicable for classification datasets)
v) Applying SVD and then running resulting dataset
vi) Compiling results from each iteration and output a results dataframe
```

2. Then, each dataset is passed through the pipeline and its results are exported into an excel sheet.
3. Results are printed on the notebook and a detailed analysis is done for the results of that dataset.

5. Implementation

5.1 Importing Libraries

In [1]:

```
#Importing Supporting Libraries
import numpy as np
import pandas as pd
from statistics import mean
import time
from numpy import *
import warnings
warnings.filterwarnings('ignore')
%load_ext autotime

#Importing Pre-processing Libraries
from pandas.api.types import is_numeric_dtype
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import  MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

#Importing Dimensionality Reduction Libraries
from sklearn.decomposition import PCA, IncrementalPCA, KernelPCA, SparsePCA, TruncatedSVD
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

#Importing Machine Learning Pipeline
from lazypredict.Supervised import LazyRegressor, LazyClassifier
import lazypredict
```

```
time: 3.75 s (started: 2022-12-30 00:51:41 +05:00)
```

5.2 Classification Datasets

5.2.1 Loading Datasets

In [8]:

```
marketing_df = pd.read_csv('Classification/MarketingDataUCI.csv', sep='\t')  #Marketing Campaign Dataset
credit_df = pd.read_csv('Classification/CreditCardUCI.csv')  #Credit Card Fraud Dataset
```

```python
heart_df = pd.read_csv('Classification/HeartDataUCI.csv')   #Heart disease Dataset
diabetic_df = pd.read_csv('Classification/DiabeticDataUCI.csv') #Diabetes Dataset
income_df = pd.read_csv('Classification/IncomeDataUCI.csv')  #High Income Prediction
beans_df = pd.read_excel('Classification/DryBeanDataUCI.xlsx')    #Drybeans dataset
banknotes_df = pd.read_csv('Classification/BankNoteAuthenticationUCI.txt',
                           header = None, names = [' variance', 'skewness',
                                                   'curtosis', 'entropy', 'Class'])  #Banknotes authentication datase
audit_df = pd.read_csv('Classification/AuditRiskUCI.csv')  #Audit Risk Dataset




#Printing Shape of each dataset
print('Shape of Marketing dataframe is: ', marketing_df.shape)
print('Shape of credit card dataframe is: ', credit_df.shape)
print('Shape of heart disease dataframe is: ', heart_df.shape)
print('Shape of diabetes dataframe is: ', diabetic_df.shape)
print('Shape of Income dataframe is: ', income_df.shape)
print('Shape of Beans dataframe is: ', beans_df.shape)
print('Shape of Bank Notes dataframe is: ', banknotes_df.shape)
print('Shape of audit dataframe is: ', audit_df.shape)
```

```
Shape of Marketing dataframe is:  (2240, 29)
Shape of credit card dataframe is:  (284807, 31)
Shape of heart disease dataframe is:  (319795, 18)
Shape of diabetes dataframe is:  (101766, 50)
Shape of Income dataframe is:  (68378, 15)
Shape of Beans dataframe is:  (13611, 17)
Shape of Bank Notes dataframe is:  (1372, 5)
Shape of audit dataframe is:  (776, 27)
time: 4.03 s (started: 2022-12-30 03:32:53 +05:00)
```

5.2.2 Pre-Processing Datasets

1. Marketing dataset

In [3]:

```python
print(marketing_df.isnull().sum()) #Check missing values
marketing_df.head()  #Bird's eye view of dataset
```

```
ID                      0
Year_Birth              0
Education               0
Marital_Status          0
Income                 24
Kidhome                 0
Teenhome                0
Dt_Customer             0
Recency                 0
MntWines                0
MntFruits               0
MntMeatProducts         0
MntFishProducts         0
MntSweetProducts        0
MntGoldProds            0
NumDealsPurchases       0
NumWebPurchases         0
NumCatalogPurchases     0
NumStorePurchases       0
NumWebVisitsMonth       0
AcceptedCmp3            0
AcceptedCmp4            0
AcceptedCmp5            0
AcceptedCmp1            0
AcceptedCmp2            0
Complain                0
Z_CostContact           0
Z_Revenue               0
Response                0
dtype: int64
```

Out[3]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth
0	5524	1957	Graduation	Single	58138.00	0	0	04-09-2012	58	635	...	7
1	2174	1954	Graduation	Single	46344.00	1	1	08-03-2014	38	11	...	5
2	4141	1965	Graduation	Together	71613.00	0	0	21-08-2013	26	426	...	4
3	6182	1984	Graduation	Together	26646.00	1	0	10-02-2014	26	11	...	6
4	5324	1981	PhD	Married	58293.00	1	0	19-01-2014	94	173	...	5

5 rows × 29 columns

```
time: 15 ms (started: 2022-12-30 00:51:49 +05:00)
```

In [4]:

```
marketing_classes = marketing_df[['Response']]
marketing_df.drop(columns = ['Response', 'ID', 'Dt_Customer'], inplace = True)  #Dropping unnecessary columns

#dummy-encoding (One-hot encoding) the categorical variables
marketing_df = pd.get_dummies(marketing_df, drop_first = True)
marketing_df.shape

#Replacing missing values by Nan
imputer = SimpleImputer(missing_values=np.nan)
imputer = imputer.fit(marketing_df)
marketing_df = pd.DataFrame(imputer.transform(marketing_df), columns = (marketing_df.columns)).astype(marketing_df.dty

#Scaling and One hot Encoding
Scaler = MinMaxScaler()
marketing_df = pd.get_dummies(marketing_df)
marketing_df = pd.DataFrame(Scaler.fit_transform(marketing_df), columns = marketing_df.columns)
print('Shape of df now is: ', marketing_df.shape)
marketing_df.head()
```

```
Shape of df now is:  (2240, 35)
```

Out[4]:

	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	...	Ed
0	0.62	0.08	0.00	0.00	0.59	0.43	0.44	0.32	0.66	0.33	...	
1	0.59	0.07	0.50	0.50	0.38	0.01	0.01	0.00	0.01	0.00	...	
2	0.70	0.11	0.00	0.00	0.26	0.29	0.25	0.07	0.43	0.08	...	
3	0.88	0.04	0.50	0.00	0.26	0.01	0.02	0.01	0.04	0.01	...	
4	0.85	0.09	0.50	0.00	0.95	0.12	0.22	0.07	0.18	0.10	...	

5 rows × 35 columns

```
time: 31 ms (started: 2022-12-30 00:51:49 +05:00)
```

2. Credit Card Dataset

In [11]:

```
print(credit_df.isnull().sum()) #Check missing values
print(credit_df.dtypes)   #Check data types
credit_df.head()  #Bird's eye view of dataset
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
Time      float64
V1        float64
V2        float64
V3        float64
V4        float64
V5        float64
V6        float64
```

```
V7          float64
V8          float64
V9          float64
V10         float64
V11         float64
V12         float64
V13         float64
V14         float64
V15         float64
V16         float64
V17         float64
V18         float64
V19         float64
V20         float64
V21         float64
V22         float64
V23         float64
V24         float64
V25         float64
V26         float64
V27         float64
V28         float64
Amount      float64
Class         int64
dtype: object
```

Out[11]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.00	-1.36	-0.07	2.54	1.38	-0.34	0.46	0.24	0.10	0.36	...	-0.02	0.28	-0.11	0.07	0.13	-0.19	0.13	-0.02	149.62	0
1	0.00	1.19	0.27	0.17	0.45	0.06	-0.08	-0.08	0.09	-0.26	...	-0.23	-0.64	0.10	-0.34	0.17	0.13	-0.01	0.01	2.69	0
2	1.00	-1.36	-1.34	1.77	0.38	-0.50	1.80	0.79	0.25	-1.51	...	0.25	0.77	0.91	-0.69	-0.33	-0.14	-0.06	-0.06	378.66	0
3	1.00	-0.97	-0.19	1.79	-0.86	-0.01	1.25	0.24	0.38	-1.39	...	-0.11	0.01	-0.19	-1.18	0.65	-0.22	0.06	0.06	123.50	0
4	2.00	-1.16	0.88	1.55	0.40	-0.41	0.10	0.59	-0.27	0.82	...	-0.01	0.80	-0.14	0.14	-0.21	0.50	0.22	0.22	69.99	0

5 rows × 31 columns

```
time: 31 ms (started: 2022-12-30 03:34:58 +05:00)
```

In [6]:

```
credit_df.Class.value_counts()  #Checking Class Distribution of dataset
```

Out[6]:

```
0    284315
1       492
Name: Class, dtype: int64
time: 0 ns (started: 2022-12-25 18:33:59 +05:00)
```

In [7]:

```
credit_df = credit_df.sample(50000)  #Sampling rows from dataset
```

```
time: 16 ms (started: 2022-12-25 18:33:59 +05:00)
```

In [8]:

```
credit_classes = credit_df[['Class']]
credit_df.drop(columns = ['Time', 'Class'], inplace = True)  #Dropping unnecessary columns

#Scaling and One hot Encoding
Scaler = MinMaxScaler()
credit_df = pd.get_dummies(credit_df)
credit_df = pd.DataFrame(Scaler.fit_transform(credit_df), columns = credit_df.columns)
print('Shape of df now is: ', credit_df.shape)
credit_df.head()
```

```
Shape of df now is:  (50000, 29)
```

Out[8]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount
0	0.97	0.61	0.77	0.34	0.56	0.51	0.54	0.73	0.47	0.51	...	0.58	0.57	0.37	0.51	0.27	0.48	0.44	0.65	0.42	0.00
1	0.97	0.60	0.75	0.22	0.57	0.58	0.53	0.75	0.43	0.52	...	0.59	0.58	0.41	0.51	0.56	0.49	0.34	0.65	0.42	0.00
2	0.99	0.60	0.77	0.32	0.55	0.51	0.54	0.74	0.50	0.51	...	0.58	0.58	0.41	0.52	0.34	0.39	0.35	0.65	0.42	0.00
3	0.97	0.59	0.81	0.31	0.53	0.52	0.52	0.74	0.50	0.51	...	0.58	0.59	0.49	0.51	0.43	0.47	0.40	0.65	0.43	0.00
4	0.95	0.60	0.82	0.42	0.54	0.54	0.53	0.74	0.51	0.50	...	0.59	0.59	0.48	0.51	0.45	0.48	0.36	0.65	0.43	0.01

5 rows × 29 columns

```
time: 31 ms (started: 2022-12-25 18:33:59 +05:00)
```

3. Heart Disease Dataset

In [12]:

```
print(heart_df.isnull().sum()) #Check missing values
print(heart_df.dtypes)    #Check data types
heart_df.head()  #Bird's eye view of dataset
```

```
HeartDisease        0
BMI                 0
Smoking             0
AlcoholDrinking     0
Stroke              0
PhysicalHealth      0
MentalHealth        0
DiffWalking         0
Sex                 0
AgeCategory         0
Race                0
Diabetic            0
PhysicalActivity    0
GenHealth           0
SleepTime           0
Asthma              0
KidneyDisease       0
SkinCancer          0
dtype: int64
HeartDisease         object
BMI                 float64
Smoking              object
AlcoholDrinking      object
Stroke               object
PhysicalHealth      float64
MentalHealth        float64
DiffWalking          object
Sex                  object
AgeCategory          object
Race                 object
Diabetic             object
PhysicalActivity     object
GenHealth            object
SleepTime           float64
Asthma               object
KidneyDisease        object
SkinCancer           object
dtype: object
```

Out[12]:

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabeti
0	No	16.60	Yes	No	No	3.00	30.00	No	Female	55-59	White	Ye
1	No	20.34	No	No	Yes	0.00	0.00	No	Female	80 or older	White	No
2	No	26.58	Yes	No	No	20.00	30.00	No	Male	65-69	White	Ye
3	No	24.21	No	No	No	0.00	0.00	No	Female	75-79	White	No
4	No	23.71	No	No	No	28.00	0.00	Yes	Female	40-44	White	No

```
time: 109 ms (started: 2022-12-30 03:35:03 +05:00)
```

In [11]:

```
heart_df['HeartDisease'].value_counts()[1]/
(heart_df['HeartDisease'].value_counts()[0] +
 heart_df['HeartDisease'].value_counts()[1])   #Checking Class Distribution
```

Out[11]:

```
0.08559545959130067

time: 32 ms (started: 2022-12-25 18:33:59 +05:00)
```

In [12]:

```
heart_df = heart_df.sample(50000)  #Sampling rows from dataset
```

```
time: 31 ms (started: 2022-12-25 18:33:59 +05:00)
```

In [13]:

```
heart_classes = heart_df[['HeartDisease']]
heart_df.drop(columns = ['HeartDisease'], inplace = True)  #Dropping unnecessary columns

#Scaling and One hot Encoding
Scaler = MinMaxScaler()
heart_df = pd.get_dummies(heart_df)
heart_df = pd.DataFrame(Scaler.fit_transform(heart_df), columns = heart_df.columns)
print('Shape of df now is: ', heart_df.shape)
heart_df.head()
```

```
Shape of df now is:  (50000, 50)
```

Out[13]:

	BMI	PhysicalHealth	MentalHealth	SleepTime	Smoking_No	Smoking_Yes	AlcoholDrinking_No	AlcoholDrinking_Yes	Stroke_No	Stroke_Y
0	0.19	0.17	0.33	0.22	1.00	0.00	1.00	0.00	1.00	0.
1	0.28	0.17	0.23	0.30	1.00	0.00	1.00	0.00	1.00	0.
2	0.23	0.67	0.03	0.30	1.00	0.00	1.00	0.00	1.00	0.

	BMI	PhysicalHealth	MentalHealth	SleepTime	Smoking_No	Smoking_Yes	AlcoholDrinking_No	AlcoholDrinking_Yes	Stroke_No	Stroke_Y
3	0.29	0.00	0.00	0.30	0.00	1.00	1.00	0.00	1.00	0.
4	0.13	0.00	0.00	0.22	0.00	1.00	1.00	0.00	1.00	0.

5 rows × 50 columns

```
time: 78 ms (started: 2022-12-25 18:33:59 +05:00)
```

In [14]:

```
heart_classes.loc[
    ((heart_classes['HeartDisease'] == 'Yes')), 'HeartDisease'] = 1     #Labeling the Yes case

heart_classes.loc[
    ((heart_classes['HeartDisease'] == 'No')), 'HeartDisease'] = 0   #Labeling the No case
```

```
time: 16 ms (started: 2022-12-25 18:34:01 +05:00)
```

4. Diabetes Dataset

In [25]:

```
#Changing diag_1, diag_2 and diag_3 to numeric
diabetic_df[['diag_1', 'diag_2', 'diag_3']] = diabetic_df[['diag_1', 'diag_2', 'diag_3']].apply(pd.to_numeric, errors

#Replacing all missing values with Nan
diabetic_df = diabetic_df.replace('?', np.nan)

print(diabetic_df.isnull().sum()) #Check missing values
print(diabetic_df.dtypes)   #Check data types
diabetic_df.head()  #Bird's eye view of dataset
```

```
encounter_id                    0
patient_nbr                     0
race                         2273
gender                          0
age                             0
weight                      98569
admission_type_id               0
discharge_disposition_id        0
admission_source_id             0
time_in_hospital                0
payer_code                  40256
medical_specialty           49949
num_lab_procedures              0
num_procedures                  0
num_medications                 0
number_outpatient               0
number_emergency                0
number_inpatient                0
diag_1                       1666
diag_2                       2894
diag_3                       6481
number_diagnoses                0
max_glu_serum                   0
A1Cresult                       0
metformin                       0
repaglinide                     0
nateglinide                     0
chlorpropamide                  0
glimepiride                     0
acetohexamide                   0
glipizide                       0
glyburide                       0
tolbutamide                     0
pioglitazone                    0
rosiglitazone                   0
acarbose                        0
miglitol                        0
troglitazone                    0
tolazamide                      0
examide                         0
citoglipton                     0
insulin                         0
glyburide-metformin             0
glipizide-metformin             0
glimepiride-pioglitazone        0
metformin-rosiglitazone         0
metformin-pioglitazone          0
change                          0
diabetesMed                     0
readmitted                      0
dtype: int64
encounter_id                  int64
```

```
patient_nbr                  int64
race                        object
gender                      object
age                         object
weight                      object
admission_type_id            int64
discharge_disposition_id     int64
admission_source_id          int64
time_in_hospital             int64
payer_code                  object
medical_specialty           object
num_lab_procedures           int64
num_procedures               int64
num_medications              int64
number_outpatient            int64
number_emergency             int64
number_inpatient             int64
diag_1                     float64
diag_2                     float64
diag_3                     float64
number_diagnoses             int64
max_glu_serum               object
A1Cresult                   object
metformin                   object
repaglinide                 object
nateglinide                 object
chlorpropamide              object
glimepiride                 object
acetohexamide               object
glipizide                   object
glyburide                   object
tolbutamide                 object
pioglitazone                object
rosiglitazone               object
acarbose                    object
miglitol                    object
troglitazone                object
tolazamide                  object
examide                     object
citoglipton                 object
insulin                     object
glyburide-metformin         object
glipizide-metformin         object
glimepiride-pioglitazone    object
metformin-rosiglitazone     object
metformin-pioglitazone      object
change                      object
diabetesMed                 object
readmitted                  object
dtype: object
```

Out[25]:

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in
0	2278392	8222157	Caucasian	Female	[0-10)	NaN	6	25	1	
1	149190	55629189	Caucasian	Female	[10-20)	NaN	1	1	7	
2	64410	86047875	AfricanAmerican	Female	[20-30)	NaN	1	1	7	
3	500364	82442376	Caucasian	Male	[30-40)	NaN	1	1	7	
4	16680	42519267	Caucasian	Male	[40-50)	NaN	1	1	7	

5 rows × 50 columns

```
time: 281 ms (started: 2022-12-25 21:11:30 +05:00)
```

In [26]:

```
diabetic_df = diabetic_df.sample(50000)  #sampling rows from the dataset
```

```
time: 62 ms (started: 2022-12-25 21:11:31 +05:00)
```

In [27]:

```
#Dropping all the ID columns and columns with a lot of missing values
diabetic_classes = diabetic_df[['diabetesMed']]
diabetic_df.drop(columns = ['diabetesMed', 'encounter_id', ''  'patient_nbr', 'weight', 'admission_type_id', 'discharg
       'admission_source_id', 'payer_code', 'medical_specialty', 'encounter_id'], inplace = True)  #Dropping unnecessa

#Replacing missing values by Nan
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
imputer = imputer.fit(diabetic_df)
```

```
diabetic_df = pd.DataFrame(imputer.transform(diabetic_df), columns = (diabetic_df.columns)).astype(diabetic_df.dtypes

#dummy-encoding (One-hot encoding) the categorical variables
diabetic_df = pd.get_dummies(diabetic_df, drop_first = True)
diabetic_df.shape


#Scaling and One hot Encoding
Scaler = MinMaxScaler()
diabetic_df = pd.get_dummies(diabetic_df)
diabetic_df = pd.DataFrame(Scaler.fit_transform(diabetic_df), columns = diabetic_df.columns)
print('Shape of df now is: ', diabetic_df.shape)
diabetic_df.head()
```

```
Shape of df now is:  (50000, 77)
```

Out[27]:

	time_in_hospital	num_lab_procedures	num_procedures	num_medications	number_outpatient	number_emergency	number_inpatient	diag
0	0.15	0.00	0.00	0.16	0.00	0.00	0.00	0.
1	0.54	0.44	0.17	0.19	0.00	0.00	0.00	1.
2	0.38	0.11	0.50	0.23	0.11	0.00	0.00	0.
3	0.15	0.30	0.33	0.17	0.03	0.08	0.19	0.
4	0.77	0.40	0.50	0.33	0.00	0.00	0.00	0.

5 rows × 77 columns

```
time: 375 ms (started: 2022-12-25 21:11:34 +05:00)
```

In [28]:

```
diabetic_classes.loc[
    ((diabetic_classes['diabetesMed'] == 'Yes')), 'diabetesMed'] = 1     #Labeling the Yes case

diabetic_classes.loc[
    ((diabetic_classes['diabetesMed'] == 'No')), 'diabetesMed'] = 0   #Labeling the No case
```

```
time: 0 ns (started: 2022-12-25 21:11:37 +05:00)
```

5. Income Dataset

In [29]:

```
print(income_df.isnull().sum()) #Check missing values
print(income_df.dtypes)   #Check data types
income_df.head()  #Bird's eye view of dataset
```

```
row ID                 0
Age                    0
WorkClass              0
X1                     0
Education Level        0
X2                     0
Marital Status         0
Occupation             0
X3                     0
Gender                 0
X4                     0
X5                     0
Hours Per Week Working 0
Native Country         0
High Income            0
dtype: int64
row ID                  object
Age                    float64
WorkClass                int64
X1                     float64
Education Level          int64
X2                     float64
Marital Status           int64
Occupation               int64
X3                       int64
Gender                   int64
X4                     float64
X5                     float64
Hours Per Week Working float64
Native Country           int64
High Income              int64
dtype: object
```

Out[29]:

	row ID	Age	WorkClass	X1	Education Level	X2	Marital Status	Occupation	X3	Gender	X4	X5	Hours Per Week Working	Native Country	High Income
0	Row2	38.00	2	215646.00	1	9.00	2	2	0	0	0.00	0.00	40.00	0	0
1	Row3	53.00	2	234721.00	2	7.00	1	2	1	0	0.00	0.00	40.00	0	0
2	Row5	37.00	2	284582.00	3	14.00	1	1	0	1	0.00	0.00	40.00	0	0
3	Row7	52.00	1	209642.00	1	9.00	1	1	0	0	0.00	0.00	45.00	0	1
4	Row8	31.00	2	45781.00	3	14.00	0	3	0	1	14084.00	0.00	50.00	0	1

```
time: 16 ms (started: 2022-12-25 21:11:39 +05:00)
```

In [30]:

```
income_df = income_df.sample(40000) #sampling rows from the dataset
```

```
time: 16 ms (started: 2022-12-25 21:11:39 +05:00)
```

In [31]:

```
income_classes = income_df[['High Income']]
income_df.drop(columns = ['High Income', 'row ID'], inplace = True)  #Dropping unnecessary columns

#Scaling and One hot Encoding
Scaler = MinMaxScaler()
income_df = pd.get_dummies(income_df)
income_df = pd.DataFrame(Scaler.fit_transform(income_df), columns = income_df.columns)
print('Shape of df now is: ', income_df.shape)
income_df.head()
```

```
Shape of df now is:  (40000, 13)
```

Out[31]:

	Age	WorkClass	X1	Education Level	X2	Marital Status	Occupation	X3	Gender	X4	X5	Hours Per Week Working	Native Country
0	0.15	0.62	0.09	0.07	0.53	0.00	0.79	0.00	1.00	0.01	0.00	0.32	0.00
1	0.58	0.25	0.24	0.27	0.27	0.17	0.29	0.00	0.00	0.00	0.00	0.30	0.00
2	0.29	0.75	0.47	0.67	0.53	0.67	0.21	0.25	0.00	0.24	0.14	0.24	0.00
3	0.10	0.25	0.21	0.80	0.47	0.00	0.29	0.25	0.00	0.00	0.00	0.32	0.00
4	0.25	0.25	0.06	0.07	0.74	0.00	0.43	0.00	0.00	0.02	0.00	0.42	0.00

```
time: 16 ms (started: 2022-12-25 21:11:40 +05:00)
```

6. Dry Beans Dataset

In [308…

```
print(beans_df.isnull().sum()) #Check missing values
print(beans_df.dtypes)  #Check data types
beans_df.head()  #Bird's eye view of dataset
```

```
Area               0
Perimeter          0
MajorAxisLength    0
MinorAxisLength    0
AspectRation       0
Eccentricity       0
ConvexArea         0
EquivDiameter      0
Extent             0
Solidity           0
roundness          0
Compactness        0
ShapeFactor1       0
ShapeFactor2       0
ShapeFactor3       0
ShapeFactor4       0
Class              0
dtype: int64
Area                 int64
Perimeter          float64
MajorAxisLength    float64
MinorAxisLength    float64
AspectRation       float64
Eccentricity       float64
ConvexArea           int64
EquivDiameter      float64
Extent             float64
Solidity           float64
roundness          float64
Compactness        float64
ShapeFactor1       float64
ShapeFactor2       float64
```

```
ShapeFactor3       float64
ShapeFactor4       float64
Class               object
dtype: object
```

Out[308...

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness
0	28395	610.29	208.18	173.89	1.20	0.55	28715	190.14	0.76	0.99	0.96
1	28734	638.02	200.52	182.73	1.10	0.41	29172	191.27	0.78	0.98	0.89
2	29380	624.11	212.83	175.93	1.21	0.56	29690	193.41	0.78	0.99	0.95
3	30008	645.88	210.56	182.52	1.15	0.50	30724	195.47	0.78	0.98	0.90
4	30140	620.13	201.85	190.28	1.06	0.33	30417	195.90	0.77	0.99	0.98

```
time: 16 ms (started: 2022-12-29 01:01:20 +05:00)
```

In [309...

```
beans_classes = beans_df[['Class']]
beans_df.drop(columns = ['Class'], inplace = True)  #Dropping unnecessary columns

#Label encoding the Y variable
le = LabelEncoder()
beans_classes = pd.DataFrame(le.fit_transform(beans_classes), columns = beans_classes.columns)


#Scaling and One hot Encoding
Scaler = MinMaxScaler()
beans_df = pd.get_dummies(beans_df)
beans_df = pd.DataFrame(Scaler.fit_transform(beans_df), columns = beans_df.columns)
print('Shape of df now is: ', beans_df.shape)
beans_df.head()
```

```
Shape of df now is:  (13611, 16)
```

Out[309...

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness
0	0.03	0.06	0.04	0.15	0.12	0.48	0.03	0.07	0.67	0.92	0.93
1	0.04	0.08	0.03	0.18	0.05	0.28	0.03	0.07	0.74	0.87	0.79
2	0.04	0.07	0.05	0.16	0.13	0.50	0.04	0.08	0.72	0.93	0.91
3	0.04	0.08	0.05	0.18	0.09	0.40	0.04	0.08	0.73	0.76	0.83
4	0.04	0.07	0.03	0.20	0.03	0.17	0.04	0.08	0.70	0.95	0.99

```
time: 16 ms (started: 2022-12-29 01:01:26 +05:00)
```

7. Bank Notes Detection Dataset

In [85]:

```
print(banknotes_df.isnull().sum()) #Check missing values
print(banknotes_df.dtypes)    #Check data types
banknotes_df.head()  #Bird's eye view of dataset
```

```
 variance    0
skewness    0
curtosis    0
entropy     0
Class       0
dtype: int64
 variance    float64
skewness    float64
curtosis    float64
entropy     float64
Class         int64
dtype: object
```

Out[85]:

	variance	skewness	curtosis	entropy	Class
0	3.62	8.67	-2.81	-0.45	0
1	4.55	8.17	-2.46	-1.46	0
2	3.87	-2.64	1.92	0.11	0
3	3.46	9.52	-4.01	-3.59	0
4	0.33	-4.46	4.57	-0.99	0

```
time: 0 ns (started: 2022-12-26 01:52:46 +05:00)
```

In [86]:

```
banknotes_classes = banknotes_df[['Class']]
banknotes_df.drop(columns = ['Class'], inplace = True)  #Dropping unnecessary columns
```

```
#Scaling and One hot Encoding
Scaler = MinMaxScaler()
banknotes_df = pd.get_dummies(banknotes_df)
banknotes_df = pd.DataFrame(Scaler.fit_transform(banknotes_df), columns = banknotes_df.columns)
print('Shape of df now is: ', banknotes_df.shape)
banknotes_df.head()
```

```
Shape of df now is:  (1372, 4)
```

Out[86]:

	variance	skewness	curtosis	entropy
0	0.77	0.84	0.11	0.74
1	0.84	0.82	0.12	0.64
2	0.79	0.42	0.31	0.79
3	0.76	0.87	0.05	0.45
4	0.53	0.35	0.42	0.69

```
time: 15 ms (started: 2022-12-26 01:52:47 +05:00)
```

8. Audit Risk Dataset

In [87]:

```
print(audit_df.isnull().sum()) #Check missing values
print(audit_df.dtypes)    #Check data types
audit_df.head()  #Bird's eye view of dataset
```

```
Sector_score      0
LOCATION_ID       0
PARA_A            0
Score_A           0
Risk_A            0
PARA_B            0
Score_B           0
Risk_B            0
TOTAL             0
numbers           0
Score_B.1         0
Risk_C            0
Money_Value       1
Score_MV          0
Risk_D            0
District_Loss     0
PROB              0
RiSk_E            0
History           0
Prob              0
Risk_F            0
Score             0
Inherent_Risk     0
CONTROL_RISK      0
Detection_Risk    0
Audit_Risk        0
Risk              0
dtype: int64
Sector_score      float64
LOCATION_ID        object
PARA_A            float64
Score_A           float64
Risk_A            float64
PARA_B            float64
Score_B           float64
Risk_B            float64
TOTAL             float64
numbers           float64
Score_B.1         float64
Risk_C            float64
Money_Value       float64
Score_MV          float64
Risk_D            float64
District_Loss       int64
PROB              float64
RiSk_E            float64
History             int64
Prob              float64
Risk_F            float64
Score             float64
Inherent_Risk     float64
CONTROL_RISK      float64
Detection_Risk    float64
Audit_Risk        float64
Risk                int64
dtype: object
```

Out[87]:

	Sector_score	LOCATION_ID	PARA_A	Score_A	Risk_A	PARA_B	Score_B	Risk_B	TOTAL	numbers	...	RiSk_E	History	Prob	Risk_F	Sc
0	3.89	23	4.18	0.60	2.51	2.50	0.20	0.50	6.68	5.00	...	0.40	0	0.20	0.00	2
1	3.89	6	0.00	0.20	0.00	4.83	0.20	0.97	4.83	5.00	...	0.40	0	0.20	0.00	2
2	3.89	6	0.51	0.20	0.10	0.23	0.20	0.05	0.74	5.00	...	0.40	0	0.20	0.00	2
3	3.89	6	0.00	0.20	0.00	10.80	0.60	6.48	10.80	6.00	...	0.40	0	0.20	0.00	4
4	3.89	6	0.00	0.20	0.00	0.08	0.20	0.02	0.08	5.00	...	0.40	0	0.20	0.00	2

5 rows × 27 columns

```
time: 16 ms (started: 2022-12-26 01:52:49 +05:00)
```

In [88]:

```
audit_df = audit_df.dropna()
audit_classes = audit_df[['Risk']]
audit_df.drop(columns = ['Risk', 'LOCATION_ID'], inplace = True)  #Dropping unnecessary columns

#Scaling and One hot Encoding
Scaler = MinMaxScaler()
audit_df = pd.get_dummies(audit_df)
audit_df = pd.DataFrame(Scaler.fit_transform(audit_df), columns = audit_df.columns)
print('Shape of df now is: ', audit_df.shape)
audit_df.head()
```

```
Shape of df now is:  (775, 25)
```

Out[88]:

	Sector_score	PARA_A	Score_A	Risk_A	PARA_B	Score_B	Risk_B	TOTAL	numbers	Score_B.1	...	PROB	RiSk_E	History	Prob	Risk_F
0	0.04	0.05	1.00	0.05	0.00	0.00	0.00	0.01	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00
1	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00
2	0.04	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00
3	0.04	0.00	0.00	0.00	0.01	1.00	0.01	0.01	0.25	1.00	...	0.00	0.00	0.00	0.00	0.00
4	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00

5 rows × 25 columns

```
time: 15 ms (started: 2022-12-26 01:57:02 +05:00)
```

5.2.3 Dimensionality Reduction Techniques with ML

A) Dimensionality Reduction Pipeline for Classification datasets:

In [313…

```
def dimensionality_reduction_classification(X_train, y_train, X_test, y_test, pca_dim = 0.95, ipca_dim= 0.95, svd_dim

    print("Starting DR Pipeline...")

    print("1. Running Lazy Predict without DR")

#Running Lazy Predict without Dimensionality Reduction:
    clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric = None)
    simple_models = clf.fit(X_train, X_test, y_train, y_test)[0].sort_index()
    simple_models['dim'] = X_train.shape[1]
    simple_models.drop(columns = ['Balanced Accuracy', 'Time Taken'], inplace = True)

    print("Success!")

#Model Performances with PCA:

    #Running PCA:
    print("2. Running PCA")
    pca = PCA(n_components = pca_dim)
    pca.fit(X_train)
    X_train_transformed = pca.transform(X_train)
    X_test_transformed = pca.transform(X_test)
    print("Success!")

    #Running Lazy Predict with PCA
    print("3. Running Lazy Predict on PCA dataset")
    clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric = None)
    pcamodels = clf.fit(X_train_transformed, X_test_transformed, y_train, y_test)[0].sort_index()
    pcamodels.drop(columns = ['Balanced Accuracy', 'Time Taken'], inplace = True)
    pcamodels['dims'] = len(pca.components_)
    print("Success!")


#Model Performances with Incremental-PCA:
```

```
    #Running Incremental PCA:
    print("4. Running Incremental PCA")
    for i in range(1, X_train.shape[1], 1):
        ipca = IncrementalPCA(n_components = i)
        ipca.fit(X_train)
        X_train_transformed = ipca.transform(X_train)
        X_test_transformed = ipca.transform(X_test)

        if ipca.explained_variance_ratio_.sum() >= ipca_dim:
            print("Success!")
            print("5. Running Lazy Predict on Incremental PCA dataset")
            clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric = None)     #Running Lazy Predict afte
            ipcamodels = clf.fit(X_train_transformed, X_test_transformed, y_train, y_test)[0].sort_index()
            ipcamodels.drop(columns = ['Balanced Accuracy', 'Time Taken'], inplace = True)
            ipcamodels['dims'] = ipca.n_components_
            print("Success!")
            break

#Model Performances with Sparse-PCA:
    print("6. Running Sparse PCA")
    spca = SparsePCA(n_components = 10)
    spca.fit(X_train)
    X_train_transformed = spca.transform(X_train)
    X_test_transformed = spca.transform(X_test)
    print("Success!")

    #Running Lazy Predict Sparse PCA:
    print("7. Running Lazy Predict on Sparse PCA dataset")
    clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric = None)     #Running Lazy Predict after SVD
    spcamodels = clf.fit(X_train_transformed, X_test_transformed, y_train, y_test)[0].sort_index()
    spcamodels.drop(columns = ['Balanced Accuracy', 'Time Taken'], inplace = True)
    spcamodels['dims'] = spca.n_components_
    print("Success!")

#Model Performances with LDA:

    #Running LDA:
    print("8. Running LDA")
    lda = LinearDiscriminantAnalysis()
    lda.fit(X_train, y_train)
    X_train_transformed = lda.transform(X_train)
    X_test_transformed = lda.transform(X_test)
    print("Success!")

    #Running Lazy Predict with LDA
    print("9. Running Lazy Predict on LDA dataset")
    clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric = None)
    ldamodels = clf.fit(X_train_transformed, X_test_transformed, y_train, y_test)[0].sort_index()
    ldamodels.drop(columns = ['Balanced Accuracy', 'Time Taken'], inplace = True)
    ldamodels['dims'] = len(lda.coef_)
    print("Success!")
#Model Performances with SVD:

    #Running SVD:
    print("10. Running SVD")
    for i in range(1, X_train.shape[1], 1):
        svd = TruncatedSVD(n_components = i)
        svd.fit(X_train)
        X_train_transformed = svd.transform(X_train)
        X_test_transformed = svd.transform(X_test)

        if svd.explained_variance_ratio_.sum() >= svd_dim or i>=X_train.shape[1]-1:
            print("Success!")
            print("11. Running Lazy Predict on SVD dataset")
            clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric = None)     #Running Lazy Predict afte
            svdmodels = clf.fit(X_train_transformed, X_test_transformed, y_train, y_test)[0].sort_index()
            svdmodels.drop(columns = ['Balanced Accuracy', 'Time Taken'], inplace = True)
            svdmodels['dims'] = len(svd.components_)
            print("Success!")
            break



#Compiling Model Results:
    print("Compiling Model Results")
    models_results = pd.concat([simple_models,
                                pcamodels,
                                ipcamodels,
                                spcamodels,
                                ldamodels,
                                svdmodels], axis = 1, keys =['Without DR', 'PCA ', 'Incremental-PCA', 'Sparse-PCA', 'I

    print("Pipeline run Successful")
```

```
time: 0 ns (started: 2022-12-29 01:02:04 +05:00)
```

B) Applying pipeline to classification datasets

1. Marketing Dataset

In [5]:

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(marketing_df, marketing_classes, test_size=0.25, random_state =43
```

```
time: 0 ns (started: 2022-12-30 00:51:57 +05:00)
```

In [294…

```
#Pipeline run
models_results = dimensionality_reduction_classification(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = (

#keeping only the desired algorithms
results = models_results.T[['AdaBoostClassifier', 'BernoulliNB',
       'DecisionTreeClassifier', 'KNeighborsClassifier', 'LinearSVC', 'LogisticRegression',
       'RandomForestClassifier', 'XGBClassifier']].T

#Exporting results to excel
results.to_excel('Results/Classification/Marketing.xlsx', sheet_name = 'Marketing Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:01<00:00, 16.11it/
s]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:02<00:00, 14.34it/
s]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:02<00:00, 14.32it/
s]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:01<00:00, 22.22it/
s]
Success!
8. Running LDA
Success!
9. Running Lazy Predict on LDA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:00<00:00, 29.14it/
s]
Success!
10. Running SVD
Success!
11. Running Lazy Predict on SVD dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:02<00:00, 14.04it/
s]
Success!
Compiling Model Results
Pipeline run Successful
time: 11.5 s (started: 2022-12-29 00:51:12 +05:00)
```

Dataset Results:

In [13]:

```
#Results
results = pd.read_excel('Results/Classification/Marketing.xlsx', header=[0, 1], index_col=0)
results
```

Out[13]:

	Without DR				PCA				Incremental-PCA		...	Sparse-PCA				
	Accuracy	ROC AUC	F1 Score	dim	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	...	F1 Score	dims	Accuracy	ROC AUC	F1 Score
Model																
AdaBoostClassifier	0.87	0.68	0.85	35	0.85	0.63	0.83	18	0.87	0.66	...	0.84	10	0.86	0.68	0.85
BernoulliNB	0.78	0.68	0.79	35	0.84	0.56	0.79	18	0.84	0.58	...	0.84	10	0.83	0.50	0.75
DecisionTreeClassifier	0.84	0.70	0.84	35	0.84	0.67	0.83	18	0.82	0.65	...	0.80	10	0.83	0.66	0.82

	Without DR				PCA				Incremental-PCA		...	Sparse-PCA				
	Accuracy	ROC AUC	F1 Score	dim	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	...	F1 Score	dims	Accuracy	ROC AUC	F1 Score
Model																
KNeighborsClassifier	0.85	0.61	0.82	35	0.84	0.61	0.82	18	0.84	0.61	...	0.83	10	0.85	0.66	0.83
LinearSVC	0.86	0.65	0.84	35	0.87	0.65	0.85	18	0.88	0.65	...	0.84	10	0.87	0.64	0.84
LogisticRegression	0.88	0.68	0.86	35	0.87	0.66	0.85	18	0.87	0.66	...	0.86	10	0.86	0.64	0.84
RandomForestClassifier	0.86	0.62	0.83	35	0.86	0.64	0.83	18	0.86	0.63	...	0.84	10	0.83	0.66	0.82
XGBClassifier	0.88	0.71	0.87	35	0.87	0.69	0.85	18	0.86	0.66	...	0.84	10	0.86	0.65	0.84

8 rows × 24 columns

```
time: 31 ms (started: 2022-12-31 02:39:22 +05:00)
```

Analysis:

1. For the marketing dataset, initially with full dataset having 35 features gave us the maximum accuracy of 0.88 and F1 score of 0.87 from the XG Boost Algorithm. For Naive bayes, we saw an accuracy of 0.78 and F1 score of 0.79.
2. After applying normal PCA, the accuracy and F1 score are almost the same as without applying PCA which means that almost all variation was captured by the Principal Components. Furthermore, Naive Bayes algorithm performance increased greatly when feature reduction was applied as its performance increased from 0.76 to 0.84. However, it is important to note that AUC-ROC of the algorithm dropped.
3. Applying the other PCA variants such as Incremental and Sparse PCA does not improve the results and they perform at Par with PCA although sparse PCA reduced the dimensions further to 10 with truly little sacrifice to variance capture. Naive Bayes algorithm's performance increased further as well. However, it is important to note that AUC-ROC of the algorithm dropped.
4. Then Linear Discriminant Analysis was tried, and the performance was at Par with the other DR techniques. However, since it reduces the dimensions to 1, it proves to be the best DR technique so far.
5. Lastly, Singular Value Decomposition also performed at PAR with the other DR techniques and gave promising results.

From this dataset's perspective LDA performs well since it reduces the dimensions to just 1 with truly little compromise on variance capture.

2. Credit Card Dataset

In [24]:

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(credit_df, credit_classes, test_size=0.25, random_state =43)
```

```
time: 16 ms (started: 2022-12-25 18:35:54 +05:00)
```

In [25]:

```
#Pipeline run
models_results = dimensionality_reduction_classification(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = (

#keeping only the desired algorithms
results = models_results.T[['AdaBoostClassifier', 'BernoulliNB',
        'DecisionTreeClassifier', 'KNeighborsClassifier', 'LinearSVC', 'LogisticRegression',
        'RandomForestClassifier', 'XGBClassifier']].T

#Exporting results to excel
results.to_excel('Results/Classification/credit.xlsx', sheet_name = 'Credit Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████| 29/29 [03:10<00:00,  6.56s/i
t]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████| 29/29 [02:45<00:00,  5.72s/i
t]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████| 29/29 [03:05<00:00,  6.38s/i
t]
Success!
6. Running Sparse PCA
```

```
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|████████████████████████████████████████| 29/29 [03:26<00:00,  7.12s/i
t]
Success!
8. Running LDA
Success!
9. Running Lazy Predict on LDA dataset
100%|████████████████████████████████████████| 29/29 [02:36<00:00,  5.41s/i
t]
Success!
10. Running SVD
Success!
11. Running Lazy Predict on SVD dataset
100%|████████████████████████████████████████| 29/29 [03:02<00:00,  6.29s/i
t]
Success!
Compiling Model Results
Pipeline run Successful
time: 18min 9s (started: 2022-12-25 18:35:55 +05:00)
```

Dataset Results:

In [299…

```
#Results
results = pd.read_excel('Results/Classification/credit.xlsx', header=[0, 1], index_col=0 )
results
```

Out[299…

	Without DR				PCA				Incremental-PCA				Sparse-PC.			
	Accuracy	ROC AUC	F1 Score	dim	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dim
Model																
AdaBoostClassifier	1.00	0.87	1.00	29	1.00	0.83	1.00	18	1.00	0.83	1.00	19	1.00	0.79	1.00	1
BernoulliNB	1.00	0.83	1.00	29	1.00	0.62	1.00	18	1.00	0.67	1.00	19	1.00	0.50	1.00	1
DecisionTreeClassifier	1.00	0.87	1.00	29	1.00	0.87	1.00	18	1.00	0.87	1.00	19	1.00	0.83	1.00	1
KNeighborsClassifier	1.00	0.79	1.00	29	1.00	0.87	1.00	18	1.00	0.87	1.00	19	1.00	0.71	1.00	1
LinearSVC	1.00	0.62	1.00	29	1.00	0.71	1.00	18	1.00	0.75	1.00	19	1.00	0.62	1.00	1
LogisticRegression	1.00	0.67	1.00	29	1.00	0.67	1.00	18	1.00	0.67	1.00	19	1.00	0.71	1.00	1
RandomForestClassifier	1.00	0.83	1.00	29	1.00	0.87	1.00	18	1.00	0.83	1.00	19	1.00	0.79	1.00	1
XGBClassifier	1.00	0.83	1.00	29	1.00	0.87	1.00	18	1.00	0.87	1.00	19	1.00	0.79	1.00	1

```
time: 32 ms (started: 2022-12-29 00:58:08 +05:00)
```

Analysis:

DR Techniques were tried on Credit Card dataset and the results are summarized below:

1. Accuracy and F1-measure of this dataset cannot be compared because it is remarkably close to 1. This is because the dataset is highly imbalanced with 0.15% of fraud classes. Therefore AUC-ROC is a better metric to gauge for this dataset.
2. The best AUC-ROC achieved without any DR technique was 0.87 which is kept as a benchmark to compare.
3. When normal PCA is applied, some algorithm's AUC-ROC increased such as Support Vector Machine and Random Forest. The best AUC-ROC remained the same at 0.87 which showed PCA to be a good technique capturing all the variance.
4. Then, when more PCA variants were tried, the performance was at par for incremental PCA but for Sparse PCA, the AUC-ROC dropped which shows that reducing the dimensions too much to 10 compromises on variance capture.
5. However, for LDA on this dataset, the performance was at par with full dataset which makes it the best DR technique as far as it reduced the dimension to 1.
6. For SVD, some algorithms suffered dips, but most algorithms performed at par with the without DR which proves it is a good technique as well.

It is evident that LDA has performed good for this dataset as it captures all variance while reducing the total dimensions to 1.

3. Heart Disease Dataset

In [26]:

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(heart_df, heart_classes.astype('int'), test_size=0.25, random_sta
```

```
time: 46 ms (started: 2022-12-25 18:55:49 +05:00)
```

In [27]:

```
#Pipeline run
models_results = dimensionality_reduction_classification(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = (
```

```python
#keeping only the desired algorithms
results = models_results.T[['AdaBoostClassifier', 'BernoulliNB',
       'DecisionTreeClassifier', 'KNeighborsClassifier', 'LinearSVC', 'LogisticRegression',
       'RandomForestClassifier', 'XGBClassifier']].T

#Exporting results to excel
results.to_excel('Results/Classification/heart.xlsx', sheet_name = 'Heart Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████| 29/29 [04:29<00:00,  9.29s/i
t]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████| 29/29 [04:24<00:00,  9.12s/i
t]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████| 29/29 [04:18<00:00,  8.92s/i
t]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|██████████████████████████████████████████████████| 29/29 [03:32<00:00,  7.34s/i
t]
Success!
8. Running LDA
Success!
9. Running Lazy Predict on LDA dataset
100%|██████████████████████████████████████████████████| 29/29 [02:58<00:00,  6.16s/i
t]
Success!
10. Running SVD
Success!
11. Running Lazy Predict on SVD dataset
100%|██████████████████████████████████████████████████| 29/29 [04:25<00:00,  9.14s/i
t]
Success!
Compiling Model Results
Pipeline run Successful
time: 24min 22s (started: 2022-12-25 18:55:49 +05:00)
```

Dataset Results:

In [300…

```python
#Results
results = pd.read_excel('Results/Classification/heart.xlsx', header=[0, 1], index_col=0 )
results
```

Out[300…

	Without DR				PCA				Incremental-PCA				Sparse-PC.			
	Accuracy	ROC AUC	F1 Score	dim	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dim
Model																
AdaBoostClassifier	0.91	0.56	0.89	50	0.91	0.55	0.89	28	0.91	0.55	0.89	28	0.91	0.55	0.89	1
BernoulliNB	0.86	0.70	0.87	50	0.91	0.54	0.88	28	0.91	0.54	0.88	28	0.90	0.58	0.89	1
DecisionTreeClassifier	0.86	0.58	0.86	50	0.87	0.59	0.87	28	0.87	0.60	0.87	28	0.87	0.60	0.87	1
KNeighborsClassifier	0.90	0.56	0.89	50	0.90	0.56	0.88	28	0.90	0.55	0.88	28	0.90	0.55	0.88	1
LinearSVC	0.92	0.52	0.88	50	0.92	0.52	0.88	28	0.92	0.51	0.88	28	0.91	0.50	0.87	1
LogisticRegression	0.92	0.55	0.89	50	0.91	0.55	0.89	28	0.91	0.55	0.89	28	0.91	0.53	0.88	1
RandomForestClassifier	0.91	0.55	0.88	50	0.90	0.55	0.88	28	0.90	0.55	0.88	28	0.90	0.56	0.88	1
XGBClassifier	0.91	0.55	0.89	50	0.91	0.55	0.89	28	0.91	0.55	0.89	28	0.91	0.54	0.89	1

```
time: 31 ms (started: 2022-12-29 00:59:21 +05:00)
```

Analysis:

Since the heart disease dataset is slightly imbalanced, we can use AUC-ROC or F1 score to gauge its performance. I would be making comparison based on both of these.

1. Without DR, the best F1-score is achieved to be 0.89 by several algorithms with an AUC-ROC of 0.56. This would be used as a benchmark for further comparisons.
2. When PCA is applied, the F1-score and AUC-ROC were still the same as without DR which proved to be a good DR technique as it reduces the dims from 50 to 28.
3. Similarly, for other PCA variants, they all performed at PAR however, sparse PCA performs better in a way that it does not compromise on variance capture and reduces dimensions the most which is 10.
4. When LDA is applied, there is barely any compromise on variance capture as the metrics remain same however, the dimensions decrease to 1 which is its major achievement.
5. SVD performs at par with the other techniques however, the dims are still too high compared to the others.

It is becoming evident that LDA is the winner DR technique!

4. Diabetes Dataset

In [32]:

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(diabetic_df, diabetic_classes.astype('int'), test_size=0.25, rand
```

```
time: 16 ms (started: 2022-12-25 21:11:50 +05:00)
```

In [33]:

```
#Pipeline run
models_results = dimensionality_reduction_classification(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = (

#keeping only the desired algorithms
results = models_results.T[['AdaBoostClassifier', 'BernoulliNB',
       'DecisionTreeClassifier', 'KNeighborsClassifier', 'LinearSVC', 'LogisticRegression',
       'RandomForestClassifier', 'XGBClassifier']].T

#Exporting results to excel
results.to_excel('Results/Classification/Diabetes.xlsx', sheet_name = 'Diabetes Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████| 29/29 [03:43<00:00,  7.72s/i
t]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████| 29/29 [03:19<00:00,  6.87s/i
t]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████| 29/29 [03:19<00:00,  6.89s/i
t]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|██████████████████████████████████████████████████| 29/29 [03:00<00:00,  6.23s/i
t]
Success!
8. Running LDA
Success!
9. Running Lazy Predict on LDA dataset
100%|██████████████████████████████████████████████████| 29/29 [03:03<00:00,  6.33s/i
t]
Success!
10. Running SVD
Success!
11. Running Lazy Predict on SVD dataset
100%|██████████████████████████████████████████████████| 29/29 [03:23<00:00,  7.03s/i
t]
Success!
Compiling Model Results
Pipeline run Successful
time: 20min 51s (started: 2022-12-25 21:11:50 +05:00)
```

Dataset Results:

In [301...

```
#Results
results = pd.read_excel('Results/Classification/diabetes.xlsx', header=[0, 1], index_col=0 )
results
```

Out[301…

	Without DR				PCA				Incremental-PCA				Sparse-PC			
	Accuracy	ROC AUC	F1 Score	dim	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dim
Model																
AdaBoostClassifier	1.00	1.00	1.00	77	1.00	1.00	1.00	30	1.00	1.00	1.00	30	0.99	0.99	0.99	1
BernoulliNB	1.00	1.00	1.00	77	0.96	0.95	0.96	30	0.94	0.90	0.93	30	0.93	0.95	0.94	1
DecisionTreeClassifier	1.00	1.00	1.00	77	1.00	1.00	1.00	30	1.00	0.99	1.00	30	0.98	0.98	0.98	1
KNeighborsClassifier	0.99	0.99	0.99	77	0.99	0.99	0.99	30	0.99	0.99	0.99	30	0.99	0.99	0.99	1
LinearSVC	1.00	1.00	1.00	77	1.00	1.00	1.00	30	1.00	1.00	1.00	30	0.99	0.99	0.99	1
LogisticRegression	1.00	1.00	1.00	77	1.00	1.00	1.00	30	1.00	1.00	1.00	30	0.99	0.99	0.99	1
RandomForestClassifier	1.00	1.00	1.00	77	1.00	1.00	1.00	30	1.00	1.00	1.00	30	0.99	0.99	0.99	1
XGBClassifier	1.00	1.00	1.00	77	1.00	1.00	1.00	30	1.00	1.00	1.00	30	0.99	0.99	0.99	1

```
time: 15 ms (started: 2022-12-29 00:59:33 +05:00)
```

Analysis:

There cannot be much analysis done on this dataset because all its values are 1 or close to 1 which is too good to be true. It is possible that the sub sample of dataset taken may have very few positive labels which would make it a highly imbalanced dataset and more prone to predicting the same class. But, in any case, it can be noticed that all the DR Techniques perform the same as non-DR one and LDA reduces the dimensions the most, so it still is the winner DR technique.

5. Income Dataset

In [37]:

```python
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(income_df, income_classes.astype('int'), test_size=0.25, random_st
```

```
time: 0 ns (started: 2022-12-25 21:55:31 +05:00)
```

In [38]:

```python
#Pipeline run
models_results = dimensionality_reduction_classification(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = (

#keeping only the desired algorithms
results = models_results.T[['AdaBoostClassifier', 'BernoulliNB',
       'DecisionTreeClassifier', 'KNeighborsClassifier', 'LinearSVC', 'LogisticRegression',
       'RandomForestClassifier', 'XGBClassifier']].T

#Exporting results to excel
results.to_excel('Results/Classification/Income.xlsx', sheet_name = 'Income Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████████████████████| 29/29 [02:53<00:00,  5.99s/i
t]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [02:58<00:00,  6.15s/i
t]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [03:01<00:00,  6.25s/i
t]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [02:48<00:00,  5.80s/i
t]
Success!
8. Running LDA
Success!
9. Running Lazy Predict on LDA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [02:17<00:00,  4.75s/i
t]
Success!
10. Running SVD
Success!
11. Running Lazy Predict on SVD dataset
```

```
100%|████████████████████████████████████████████████████████████████████| 29/29 [03:00<00:00,  6.23s/i
t]
Success!
Compiling Model Results
Pipeline run Successful
time: 17min 4s (started: 2022-12-25 21:55:32 +05:00)
```

Dataset Results:

In [302…

```
#Results
results = pd.read_excel('Results/Classification/income.xlsx', header=[0, 1], index_col=0 )
results
```

Out[302…

	Without DR				PCA				Incremental-PCA				Sparse-PC.			
	Accuracy	ROC AUC	F1 Score	dim	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dim
Model																
AdaBoostClassifier	0.75	0.69	0.74	13	0.72	0.64	0.71	12	0.71	0.63	0.70	12	0.74	0.67	0.73	1
BernoulliNB	0.69	0.63	0.68	13	0.70	0.57	0.65	12	0.69	0.56	0.64	12	0.69	0.62	0.68	1
DecisionTreeClassifier	0.75	0.72	0.75	13	0.72	0.68	0.72	12	0.72	0.68	0.72	12	0.73	0.69	0.73	1
KNeighborsClassifier	0.74	0.69	0.74	13	0.74	0.69	0.74	12	0.74	0.69	0.74	12	0.74	0.69	0.73	1
LinearSVC	0.69	0.57	0.65	13	0.69	0.57	0.65	12	0.69	0.57	0.65	12	0.69	0.57	0.65	1
LogisticRegression	0.69	0.57	0.65	13	0.69	0.57	0.65	12	0.69	0.57	0.65	12	0.69	0.57	0.65	1
RandomForestClassifier	0.78	0.75	0.78	13	0.76	0.70	0.75	12	0.77	0.71	0.76	12	0.76	0.72	0.76	1
XGBClassifier	0.77	0.73	0.77	13	0.75	0.70	0.74	12	0.76	0.71	0.75	12	0.75	0.71	0.75	1

```
time: 15 ms (started: 2022-12-29 00:59:52 +05:00)
```

Analysis:
The dataset is not that imbalanced so we can use any of the three metrics to compare the DR techniques. To keep it coherent, lets focus on F1 score for this dataset:

1. The best F1 score achieved with full dataset without any DR technique is around 0.78 given by Random Forest.
2. After applying PCA, a very slight drop in F1 score is seen however, the dims are not reduced that much either and remain at 12 reduced from 13. So, at the compromise of little variance capture, only 1 feature is reduced.
3. Applying different variants of PCA resulted in comparable results. However, sparse PCA performs better in a way that it reduces dimensions from 12 to 10 with truly little compromise on variance capture.
4. After applying LDA, the features are reduced to 1 however, there is a significant drop in variance capture which shows that LDA does not perform good on this dataset.
5. Lastly, SVD performed like PCA.

It is seen that in this dataset, DR techniques cause a significant decrease in variance capture if features are reduced to less and the conclusion reached from this is that the dataset already contains the most important information.

6. Dry Beans Dataset

In [314…

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(beans_df, beans_classes.astype('int'), test_size=0.25, random_stat
```

```
time: 0 ns (started: 2022-12-29 01:02:12 +05:00)
```

In [315…

```
#Pipeline run
models_results = dimensionality_reduction_classification(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = (

#keeping only the desired algorithms
results = results[[]]
results = models_results.T[['AdaBoostClassifier', 'BernoulliNB',
       'DecisionTreeClassifier', 'KNeighborsClassifier', 'LinearSVC', 'LogisticRegression',
       'RandomForestClassifier', 'XGBClassifier']].T

#Exporting results to excel
results.to_excel('Results/Classification/Beans.xlsx', sheet_name = 'Beans Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|████████████████████████████████████████████████████████████████████| 29/29 [00:19<00:00,  1.48it/
s]
Success!
2. Running PCA
```

```
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████████████████████████████████| 29/29 [00:15<00:00,  1.92it/
s]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████████████████████████████████| 29/29 [00:14<00:00,  1.94it/
s]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|██████████████████████████████████████████████████████████████████████████████| 29/29 [00:16<00:00,  1.77it/
s]
Success!
8. Running LDA
Success!
9. Running Lazy Predict on LDA dataset
100%|██████████████████████████████████████████████████████████████████████████████| 29/29 [00:12<00:00,  2.24it/
s]
Success!
10. Running SVD
Success!
11. Running Lazy Predict on SVD dataset
100%|██████████████████████████████████████████████████████████████████████████████| 29/29 [00:14<00:00,  1.97it/
s]
Success!
Compiling Model Results
Pipeline run Successful
time: 1min 34s (started: 2022-12-29 01:02:13 +05:00)
```

Dataset Results:

In [332…

```
#Results
results = pd.read_excel('Results/Classification/Beans.xlsx', header=[0, 1], index_col=0 )
results
```

Out[332…

	Without DR			PCA			Incremental-PCA			Sparse-PCA			LDA		
	Accuracy	F1 Score	dim	Accuracy	F1 Score	dims	Accuracy	F1 Score	dims	Accuracy	F1 Score	dims	Accuracy	F1 Score	dim
Model															
AdaBoostClassifier	0.68	0.62	16	0.44	0.27	4	0.44	0.27	4	0.74	0.70	10	0.73	0.72	
BernoulliNB	0.72	0.72	16	0.65	0.58	4	0.66	0.58	4	0.77	0.76	10	0.85	0.85	
DecisionTreeClassifier	0.89	0.89	16	0.85	0.85	4	0.85	0.85	4	0.89	0.89	10	0.89	0.89	
KNeighborsClassifier	0.93	0.93	16	0.89	0.89	4	0.89	0.89	4	0.93	0.93	10	0.92	0.92	
LinearSVC	0.91	0.91	16	0.85	0.85	4	0.86	0.85	4	0.91	0.91	10	0.91	0.91	
LogisticRegression	0.92	0.92	16	0.89	0.89	4	0.89	0.89	4	0.92	0.92	10	0.92	0.92	
RandomForestClassifier	0.92	0.92	16	0.89	0.89	4	0.89	0.89	4	0.93	0.93	10	0.93	0.93	
XGBClassifier	0.92	0.92	16	0.89	0.89	4	0.89	0.89	4	0.92	0.92	10	0.92	0.92	

```
time: 15 ms (started: 2022-12-29 01:16:14 +05:00)
```

Analysis:

For the dry bean's dataset, AUC-ROC was removed since it is a multi-class problem. Since the dataset is balanced, I will be focusing on Accuracy as the metric for comparison.

1. The best accuracy achieved for this dataset without any DR technique is 0.92.
2. After applying PCA, the accuracy slightly decreased but the dimensions went from 16 to 4 which is good.
3. After applying variants of PCA, the accuracy remained at Par with normal PCA. However, sparse PCA gave a slightly better accuracy of 0.93 and reduced features to 10 from 16.
4. After applying LDA, the dimensions were reduced to 7 which is exceptionally good with no expense to accuracy and variance capture. LDA proves to be good so far.
5. SVD reduced dimensions to 4 but there was a slight drop in accuracy.

LDA still proves to be good as it maintains the best accuracy with significant reduction of features. The other DR techniques are still good enough

7. Bank Notes Dataset

In [76]:
```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(banknotes_df, banknotes_classes.astype('int'), test_size=0.25, ra
```

```
time: 0 ns (started: 2022-12-26 01:51:27 +05:00)
```

In [77]:
```
#Pipeline run
models_results = dimensionality_reduction_classification(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = (

#keeping only the desired algorithms
results = models_results.T[['AdaBoostClassifier', 'BernoulliNB',
       'DecisionTreeClassifier', 'KNeighborsClassifier', 'LinearSVC', 'LogisticRegression',
       'RandomForestClassifier', 'XGBClassifier']].T

#Exporting results to excel
results.to_excel('Results/Classification/BankNotes.xlsx', sheet_name = 'Bank Notes Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|████████████████████████████████████████████████████| 29/29 [00:00<00:00, 43.32it/
s]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|████████████████████████████████████████████████████| 29/29 [00:00<00:00, 37.63it/
s]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|████████████████████████████████████████████████████| 29/29 [00:00<00:00, 37.69it/
s]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|████████████████████████████████████████████████████| 29/29 [00:00<00:00, 37.18it/
s]
Success!
8. Running LDA
Success!
9. Running Lazy Predict on LDA dataset
100%|████████████████████████████████████████████████████| 29/29 [00:00<00:00, 43.57it/
s]
Success!
10. Running SVD
Success!
11. Running Lazy Predict on SVD dataset
100%|████████████████████████████████████████████████████| 29/29 [00:00<00:00, 37.38it/
s]
Success!
Compiling Model Results
Pipeline run Successful
time: 4.62 s (started: 2022-12-26 01:51:28 +05:00)
```

Dataset Results:

In [333…
```
#Results
results = pd.read_excel('Results/Classification/BankNotes.xlsx', header=[0, 1], index_col=0 )
results
```

Out[333…

	Without DR				PCA				Incremental-PCA				Sparse-PC.			
	Accuracy	ROC AUC	F1 Score	dim	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dim
Model																
AdaBoostClassifier	0.99	0.99	0.99	4	0.92	0.91	0.92	3	0.93	0.93	0.93	3	0.99	0.99	0.99	1
BernoulliNB	0.86	0.86	0.86	4	0.79	0.77	0.78	3	0.79	0.77	0.78	3	0.86	0.86	0.86	1
DecisionTreeClassifier	0.98	0.98	0.98	4	0.95	0.95	0.95	3	0.94	0.94	0.94	3	0.99	0.99	0.99	1
KNeighborsClassifier	1.00	1.00	1.00	4	0.97	0.97	0.97	3	0.97	0.97	0.97	3	1.00	1.00	1.00	1
LinearSVC	0.99	0.99	0.99	4	0.92	0.91	0.92	3	0.92	0.92	0.92	3	0.99	0.99	0.99	1
LogisticRegression	0.98	0.98	0.98	4	0.92	0.92	0.92	3	0.92	0.92	0.92	3	0.98	0.98	0.98	1
RandomForestClassifier	0.99	0.99	0.99	4	0.95	0.95	0.95	3	0.95	0.95	0.95	3	0.99	0.99	0.99	1

	Without DR				PCA				Incremental-PCA				Sparse-PC.			
	Accuracy	ROC AUC	F1 Score	dim	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dim
Model																
XGBClassifier	0.99	0.99	0.99	4	0.94	0.95	0.94	3	0.94	0.95	0.94	3	0.99	0.99	0.99	1

```
time: 16 ms (started: 2022-12-29 01:20:27 +05:00)
```

Analysis:

This dataset is a banknote authentication dataset. Since the dataset is balanced, we can use accuracy as a metric to gauge it:

1. The accuracy achieved without DR techniques is 1 which seems that the model is too good to be true.
2. After applying PCA, the accuracy is slightly compromised to 0.97 with just 1 feature reduction.
3. Different PCA variants perform similar with sparse PCA performing worse as it increases the dimensions from 4 to 10.
4. LDA captures all the variance of the dataset and does not compromise on accuracy while reducing the number of dimensions from 4 to 1 which shows that it performs very well.
5. SVD performs at par with PCA.

It can be seen here as well that LDA performs well for this dataset as it reduces number of dimensions to 1 without any compromise in accuracy.

8. Audit Risk Dataset

In [89]:

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(audit_df, audit_classes.astype('int'), test_size=0.25, random_stat
```

```
time: 0 ns (started: 2022-12-26 01:57:13 +05:00)
```

In [90]:

```
#Pipeline run
models_results = dimensionality_reduction_classification(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = (

#keeping only the desired algorithms
results = models_results.T[['AdaBoostClassifier', 'BernoulliNB',
       'DecisionTreeClassifier', 'KNeighborsClassifier', 'LinearSVC', 'LogisticRegression',
       'RandomForestClassifier', 'XGBClassifier']].T

#Exporting results to excel
results.to_excel('Results/Classification/AuditRisk.xlsx', sheet_name = 'Audit Risk Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:00<00:00, 59.52it/
s]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:00<00:00, 47.02it/
s]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:00<00:00, 46.61it/
s]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:00<00:00, 48.51it/
s]
Success!
8. Running LDA
Success!
9. Running Lazy Predict on LDA dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:00<00:00, 54.28it/
s]
Success!
10. Running SVD
Success!
11. Running Lazy Predict on SVD dataset
100%|██████████████████████████████████████████████████████████████████| 29/29 [00:00<00:00, 46.03it/
s]
```

```
Success!
Compiling Model Results
Pipeline run Successful
time: 3.86 s (started: 2022-12-26 01:57:14 +05:00)
```

Dataset Results:

In [335…

```
#Results
results = pd.read_excel('Results/Classification/AuditRisk.xlsx', header=[0, 1], index_col=0 )
results
```

Out[335…

	Without DR				PCA				Incremental-PCA				Sparse-PC.			
	Accuracy	ROC AUC	F1 Score	dim	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dims	Accuracy	ROC AUC	F1 Score	dim
Model																
AdaBoostClassifier	1.00	1.00	1.00	25	0.98	0.98	0.98	7	0.98	0.98	0.98	7	0.99	0.99	0.99	1
BernoulliNB	0.92	0.90	0.92	25	0.87	0.85	0.86	7	0.86	0.84	0.85	7	0.89	0.88	0.89	1
DecisionTreeClassifier	1.00	1.00	1.00	25	0.98	0.98	0.98	7	0.99	0.99	0.99	7	0.98	0.98	0.98	1
KNeighborsClassifier	0.97	0.96	0.97	25	0.96	0.95	0.96	7	0.96	0.95	0.96	7	0.96	0.95	0.96	1
LinearSVC	0.98	0.98	0.98	25	0.96	0.95	0.96	7	0.95	0.95	0.95	7	0.98	0.98	0.98	1
LogisticRegression	0.98	0.98	0.98	25	0.95	0.95	0.95	7	0.95	0.95	0.95	7	0.98	0.98	0.98	1
RandomForestClassifier	1.00	1.00	1.00	25	0.98	0.98	0.98	7	0.98	0.98	0.98	7	0.98	0.98	0.98	1
XGBClassifier	1.00	1.00	1.00	25	0.97	0.97	0.97	7	0.98	0.98	0.98	7	0.99	0.99	0.99	1

```
time: 32 ms (started: 2022-12-29 01:20:48 +05:00)
```

Analysis:

The audit risk dataset is a simple dataset therefore it has particularly good metrics. Since the dataset is balanced, I would focus on looking at accuracy for comparison:

1. The best accuracy achieved is 1.0 without applying any DR techniques. The total dims are 25.
2. After applying PCA, the accuracy slightly decreased to around 0.98 while reducing features significantly to 7 dimensions.
3. After trying other PCA variants, the results were at par with PCA with accuracy around 0.99.
4. After applying LDA, the best accuracy achieved was 0.95 which is slightly less than 1 however the number of dimensions become 1 which is a significant achievement.
5. SVD gives an accuracy of 0.98 and reduces dimensions to 7 which is the same as PCA.

All techniques perform quite well on this dataset. LDA compromises slightly on variance capture however it reduces dimensionality the most.

5.3 Regression Datasets

5.3.1 Loading Datasets

In [269…

```
power_df = pd.read_csv('Regression/CombinedCyclePowerPlantUCI.csv')  #Combined Cycle Power Plant dataset
energy_df = pd.read_excel('Regression/EnergyEfficiencyUCI.xlsx')  #Energy Efficiency Dataset
aquatic_df = pd.read_csv('Regression/qsar_aquatic_toxicityUCI.csv', sep = ';', names = ['TPSA', 'SAacc', 'H-050', 'ML(
                                                                                         'RDCHI', 'GATS1p','nN','C-040',
                                                                                          'quantitative response_LC50']]
bikes_df = pd.read_csv('Regression/SeoulBikeDataUCI.csv', encoding= 'unicode_escape')  #Bike Sharing Dataset
redwine_df = pd.read_csv('Regression/RedWineQualityUCI.csv', sep = ';')  #Red Wine Quality Dataset
student_df = pd.read_csv('Regression/student-porUCI.csv', sep = ';')  #Student Performance dataset
hardware_df = pd.read_csv('Regression/TomsHardwareUCI.data', names = ['NCD_0', 'NCD_1', 'NCD_2', 'NCD_3', 'NCD_4', 'N(
                                                                     'BL_0', 'BL_1', 'BL_2', 'BL_3', 'BL_4', 'BL_5', 'I
                                                                     'NAD_0', 'NAD_1', 'NAD_2', 'NAD_3', 'NAD_4', 'NAD_
                                                                     'AI_0', 'AI_1', 'AI_2', 'AI_3', 'AI_4', 'AI_5', '/
                                                                     'NAC_0', 'NAC_1', 'NAC_2', 'NAC_3', 'NAC_4', 'NAC_
                                                                     'ND_0', 'ND_1', 'ND_2', 'ND_3', 'ND_4', 'ND_5', 'I
                                                                     'CS_0', 'CS_1', 'CS_2', 'CS_3', 'CS_4', 'CS_5', '(
                                                                     'AT_0', 'AT_1', 'AT_2', 'AT_3', 'AT_4', 'AT_5', '/
                                                                     'NA_0', 'NA_1', 'NA_2', 'NA_3', 'NA_4', 'NA_5', 'I
                                                                     'ADL_0', 'ADL_1', 'ADL_2', 'ADL_3', 'ADL_4', 'ADL_
                                                                     'AS(NA)_0', 'AS(NA)_1', 'AS(NA)_2', 'AS(NA)_3', '/
                                                                     'AS(NAC)_0', 'AS(NAC)_1', 'AS(NAC)_2', 'AS(NAC)_3
                                                                     ])   #Tom's Hardware Dataset


print('Shape of Combined Power Plant dataframe is: ', power_df.shape)
print('Shape of Energy Efficiency dataframe is: ', energy_df.shape)
print('Shape of Aquatic Toxicity dataframe is: ', aquatic_df.shape)
```

```python
print('Shape of Seoul bike dataframe is: ', bikes_df.shape)
print('Shape of Red Wine Quality dataframe is: ', redwine_df.shape)
print('Shape of student dataframe is: ', student_df.shape)
print('Shape of Toms Hardware dataframe is: ', hardware_df.shape)
```

```
Shape of Combined Power Plant dataframe is:  (9568, 5)
Shape of Energy Efficiency dataframe is:  (768, 10)
Shape of Aquatic Toxicity dataframe is:  (546, 8)
Shape of Seoul bike dataframe is:  (8760, 13)
Shape of Red Wine Quality dataframe is:  (1599, 12)
Shape of student dataframe is:  (649, 33)
Shape of Toms Hardware dataframe is:  (28179, 97)
time: 281 ms (started: 2022-12-26 16:14:57 +05:00)
```

5.3.2 Pre-Processing Datasets

1. Power Consumption Dataset

In [254…

```python
print(power_df.isnull().sum()) #Check missing values
print(power_df.dtypes)   #Check data types
power_df.head()  #Bird's eye view of dataset
```

```
AT    0
V     0
AP    0
RH    0
PE    0
dtype: int64
AT    float64
V     float64
AP    float64
RH    float64
PE    float64
dtype: object
```

Out[254…

	AT	V	AP	RH	PE
0	8.34	40.77	1010.84	90.01	480.48
1	23.64	58.49	1011.40	74.20	445.75
2	29.74	56.90	1007.15	41.91	438.76
3	19.07	49.69	1007.22	76.79	453.09
4	11.80	40.66	1017.13	97.20	464.43

```
time: 0 ns (started: 2022-12-26 15:55:54 +05:00)
```

In [255…

```python
power_classes = power_df[['PE']]
power_df.drop(columns = ['PE'], inplace = True)  #Dropping unnecessary columns

#Scaling and One-hot encoding
Scaler = MinMaxScaler()
power_df = pd.get_dummies(power_df)
power_df = pd.DataFrame(Scaler.fit_transform(power_df), columns = power_df.columns)
print('Shape of df now is: ', power_df.shape)
power_df.head()
```

```
Shape of df now is:  (9568, 4)
```

Out[255…

	AT	V	AP	RH
0	0.18	0.27	0.44	0.86
1	0.62	0.59	0.46	0.65
2	0.79	0.56	0.35	0.22
3	0.49	0.43	0.35	0.69
4	0.28	0.27	0.60	0.96

```
time: 16 ms (started: 2022-12-26 15:55:55 +05:00)
```

2. Energy Efficiency Dataset

In [256…

```python
print(energy_df.isnull().sum()) #Check missing values
print(energy_df.dtypes)   #Check data types
energy_df.head()  #Bird's eye view of dataset
```

```
X1    0
X2    0
X3    0
X4    0
X5    0
```

```
X6    0
X7    0
X8    0
Y1    0
Y2    0
dtype: int64
X1    float64
X2    float64
X3    float64
X4    float64
X5    float64
X6      int64
X7    float64
X8      int64
Y1    float64
Y2    float64
dtype: object
```

Out[256...

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
0	0.98	514.50	294.00	110.25	7.00	2	0.00	0	15.55	21.33
1	0.98	514.50	294.00	110.25	7.00	3	0.00	0	15.55	21.33
2	0.98	514.50	294.00	110.25	7.00	4	0.00	0	15.55	21.33
3	0.98	514.50	294.00	110.25	7.00	5	0.00	0	15.55	21.33
4	0.90	563.50	318.50	122.50	7.00	2	0.00	0	20.84	28.28

```
time: 16 ms (started: 2022-12-26 15:55:56 +05:00)
```

In [257...

```
energy_classes = energy_df[['Y1']]
energy_df.drop(columns = ['Y1', 'Y2'], inplace = True)  #Dropping unnecessary columns

#Scaling and One-hot encoding
Scaler = MinMaxScaler()
energy_df = pd.get_dummies(energy_df)
energy_df = pd.DataFrame(Scaler.fit_transform(energy_df), columns = energy_df.columns)
print('Shape of df now is: ', energy_df.shape)
energy_df.head()
```

```
Shape of df now is:  (768, 8)
```

Out[257...

	X1	X2	X3	X4	X5	X6	X7	X8
0	1.00	0.00	0.29	0.00	1.00	0.00	0.00	0.00
1	1.00	0.00	0.29	0.00	1.00	0.33	0.00	0.00
2	1.00	0.00	0.29	0.00	1.00	0.67	0.00	0.00
3	1.00	0.00	0.29	0.00	1.00	1.00	0.00	0.00
4	0.78	0.17	0.43	0.11	1.00	0.00	0.00	0.00

```
time: 16 ms (started: 2022-12-26 15:55:56 +05:00)
```

3. Aquatic Toxicity Dataset

In [258...

```
print(aquatic_df.isnull().sum()) #Check missing values
print(aquatic_df.dtypes)   #Check data types
aquatic_df.head()  #Bird's eye view of dataset
```

```
TPSA                           0
SAacc                          0
H-050                          0
MLOGPRDCHI                     0
GATS1p                         0
nN                             0
C-040                          0
quantitative response_LC50     0
dtype: int64
TPSA                           float64
SAacc                            int64
H-050                          float64
MLOGPRDCHI                     float64
GATS1p                         float64
nN                               int64
C-040                            int64
quantitative response_LC50     float64
dtype: object
```

Out[258...

	TPSA	SAacc	H-050	MLOGPRDCHI	GATS1p	nN	C-040	quantitative response_LC50
0.00	0.00	0	2.42	1.23	0.67	0	0	3.74
0.00	0.00	0	2.64	1.40	0.63	0	0	4.33

	TPSA	SAacc	H-050	MLOGPRDCHI	GATS1p	nN	C-040	quantitative response_LC50
9.23	11.00	0	5.80	2.93	0.49	0	0	7.02
9.23	11.00	0	5.45	2.89	0.49	0	0	6.72
9.23	11.00	0	4.07	2.76	0.69	0	0	5.98

```
time: 15 ms (started: 2022-12-26 15:56:12 +05:00)
```

In [259…

```
aquatic_classes = aquatic_df[['quantitative response_LC50']]
aquatic_df.drop(columns = ['quantitative response_LC50'], inplace = True)  #Dropping unnecessary columns

#Scaling and One-hot encoding
Scaler = MinMaxScaler()
aquatic_df = pd.get_dummies(aquatic_df)
aquatic_df = pd.DataFrame(Scaler.fit_transform(aquatic_df), columns = aquatic_df.columns)
print('Shape of df now is: ', aquatic_df.shape)
aquatic_df.head()
```

```
Shape of df now is:  (546, 7)
```

Out[259…

	TPSA	SAacc	H-050	MLOGPRDCHI	GATS1p	nN	C-040
0	0.00	0.00	0.57	0.04	0.17	0.00	0.00
1	0.00	0.00	0.58	0.07	0.16	0.00	0.00
2	0.02	0.00	0.79	0.35	0.09	0.00	0.00
3	0.02	0.00	0.76	0.35	0.10	0.00	0.00
4	0.02	0.00	0.67	0.32	0.19	0.00	0.00

```
time: 0 ns (started: 2022-12-26 15:56:43 +05:00)
```

4. Seoul Bikes Dataset

In [195…

```
print(bikes_df.isnull().sum()) #Check missing values
print(bikes_df.dtypes)   #Check data types
bikes_df.head()  #Bird's eye view of dataset
```

```
Rented Bike Count            0
Hour                         0
Temperature(°C)              0
Humidity(%)                  0
Wind speed (m/s)             0
Visibility (10m)             0
Dew point temperature(°C)    0
Solar Radiation (MJ/m2)      0
Rainfall(mm)                 0
Snowfall (cm)                0
Seasons                      0
Holiday                      0
Functioning Day              0
dtype: int64
Rented Bike Count              int64
Hour                           int64
Temperature(°C)              float64
Humidity(%)                    int64
Wind speed (m/s)             float64
Visibility (10m)               int64
Dew point temperature(°C)    float64
Solar Radiation (MJ/m2)      float64
Rainfall(mm)                 float64
Snowfall (cm)                float64
Seasons                       object
Holiday                       object
Functioning Day               object
dtype: object
```

Out[195…

	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Fu
0	254	0	-5.20	37	2.20	2000	-17.60	0.00	0.00	0.00	Winter	No Holiday	
1	204	1	-5.50	38	0.80	2000	-17.60	0.00	0.00	0.00	Winter	No Holiday	
2	173	2	-6.00	39	1.00	2000	-17.70	0.00	0.00	0.00	Winter	No Holiday	
3	107	3	-6.20	40	0.90	2000	-17.60	0.00	0.00	0.00	Winter	No Holiday	
4	78	4	-6.00	36	2.30	2000	-18.60	0.00	0.00	0.00	Winter	No	

	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Fu
												Holiday	

```
time: 16 ms (started: 2022-12-26 03:34:54 +05:00)
```

In [196…

```
bikes_classes = bikes_df[['Rented Bike Count']]
bikes_df.drop(columns = ['Rented Bike Count'], inplace = True)  #Dropping unnecessary columns

#Scaling and One-hot encoding
Scaler = MinMaxScaler()
bikes_df = pd.get_dummies(bikes_df)
bikes_df = pd.DataFrame(Scaler.fit_transform(bikes_df), columns = bikes_df.columns)
print('Shape of df now is: ', bikes_df.shape)
bikes_df.head()
```

```
Shape of df now is:  (8760, 17)
```

Out[196…

	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons_Autumn	Seasons_Spr
0	0.00	0.22	0.38	0.30	1.00	0.22	0.00	0.00	0.00	0.00	(
1	0.04	0.22	0.39	0.11	1.00	0.22	0.00	0.00	0.00	0.00	(
2	0.09	0.21	0.40	0.14	1.00	0.22	0.00	0.00	0.00	0.00	(
3	0.13	0.20	0.41	0.12	1.00	0.22	0.00	0.00	0.00	0.00	(
4	0.17	0.21	0.37	0.31	1.00	0.21	0.00	0.00	0.00	0.00	(

```
time: 16 ms (started: 2022-12-26 03:34:54 +05:00)
```

5. Red Wine Quality Dataset

In [197…

```
print(redwine_df.isnull().sum()) #Check missing values
print(redwine_df.dtypes)   #Check data types
redwine_df.head()  #Bird's eye view of dataset
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

Out[197…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.40	0.70	0.00	1.90	0.08	11.00	34.00	1.00	3.51	0.56	9.40	5
1	7.80	0.88	0.00	2.60	0.10	25.00	67.00	1.00	3.20	0.68	9.80	5
2	7.80	0.76	0.04	2.30	0.09	15.00	54.00	1.00	3.26	0.65	9.80	5
3	11.20	0.28	0.56	1.90	0.07	17.00	60.00	1.00	3.16	0.58	9.80	6
4	7.40	0.70	0.00	1.90	0.08	11.00	34.00	1.00	3.51	0.56	9.40	5

```
time: 0 ns (started: 2022-12-26 03:34:54 +05:00)
```

In [198…

```
redwine_classes = redwine_df[['quality']]
redwine_df.drop(columns = ['quality'], inplace = True)  #Dropping unnecessary columns

#Scaling and One-hot encoding
Scaler = MinMaxScaler()
redwine_df = pd.get_dummies(redwine_df)
redwine_df = pd.DataFrame(Scaler.fit_transform(redwine_df), columns = redwine_df.columns)
print('Shape of df now is: ', redwine_df.shape)
redwine_df.head()
```

```
Shape of df now is:  (1599, 11)
```

Out[198…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	0.25	0.40	0.00	0.07	0.11	0.14	0.10	0.57	0.61	0.14	0.15
1	0.28	0.52	0.00	0.12	0.14	0.34	0.22	0.49	0.36	0.21	0.22
2	0.28	0.44	0.04	0.10	0.13	0.20	0.17	0.51	0.41	0.19	0.22
3	0.58	0.11	0.56	0.07	0.11	0.23	0.19	0.58	0.33	0.15	0.22
4	0.25	0.40	0.00	0.07	0.11	0.14	0.10	0.57	0.61	0.14	0.15

```
time: 16 ms (started: 2022-12-26 03:34:55 +05:00)
```

6. Student Portugese Dataset

In [270…

```
print(student_df.isnull().sum()) #Check missing values
print(student_df.dtypes)    #Check data types
student_df.head()  #Bird's eye view of dataset
```

```
school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
school        object
sex           object
age            int64
address       object
famsize       object
Pstatus       object
Medu           int64
Fedu           int64
Mjob          object
Fjob          object
reason        object
guardian      object
traveltime     int64
studytime      int64
failures       int64
schoolsup     object
famsup        object
paid          object
```

```
activities    object
nursery       object
higher        object
internet      object
romantic      object
famrel         int64
freetime       int64
goout          int64
Dalc           int64
Walc           int64
health         int64
absences       int64
G1             int64
G2             int64
G3             int64
dtype: object
```

Out[270…

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	4
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	2
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	6
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	0
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	0

5 rows × 33 columns

```
time: 0 ns (started: 2022-12-26 16:15:05 +05:00)
```

In [271…

```
student_classes = student_df[['G3']]
student_df.drop(columns = [ 'G3'], inplace = True)  #Dropping unnecessary columns
Scaler = MinMaxScaler()

#Scaling and One-hot encoding
student_df = pd.get_dummies(student_df)
student_df = pd.DataFrame(Scaler.fit_transform(student_df), columns = student_df.columns)
print('Shape of df now is: ', student_df.shape)
student_df.head()
```

```
Shape of df now is:  (649, 58)
```

Out[271…

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	...	activities_no	activities_yes	nursery_no	nursery_ye
0	0.43	1.00	1.00	0.33	0.33	0.00	0.75	0.50	0.75	0.00	...	1.00	0.00	0.00	1.0(
1	0.29	0.25	0.25	0.00	0.33	0.00	1.00	0.50	0.50	0.00	...	1.00	0.00	1.00	0.0(
2	0.00	0.25	0.25	0.00	0.33	0.00	0.75	0.50	0.25	0.25	...	1.00	0.00	0.00	1.0(
3	0.00	1.00	0.50	0.00	0.67	0.00	0.50	0.25	0.25	0.00	...	0.00	1.00	0.00	1.0(
4	0.14	0.75	0.75	0.00	0.33	0.00	0.75	0.50	0.25	0.00	...	1.00	0.00	0.00	1.0(

5 rows × 58 columns

```
time: 16 ms (started: 2022-12-26 16:15:14 +05:00)
```

7. Tom's Hardware Dataset

In [201…

```
print(hardware_df.isnull().sum()) #Check missing values
print(hardware_df.dtypes)   #Check data types
hardware_df.head()  #Bird's eye view of dataset
```

```
NCD_0        0
NCD_1        0
NCD_2        0
NCD_3        0
NCD_4        0
            ..
AS(NAC)_3    0
AS(NAC)_4    0
AS(NAC)_5    0
AS(NAC)_6    0
AS(NAC)_7    0
Length: 97, dtype: int64
NCD_0          int64
NCD_1          int64
NCD_2          int64
NCD_3          int64
NCD_4          int64
               ...
```

```
AS(NAC)_3    float64
AS(NAC)_4    float64
AS(NAC)_5    float64
AS(NAC)_6    float64
AS(NAC)_7    float64
Length: 97, dtype: object
```

Out[201…

	NCD_0	NCD_1	NCD_2	NCD_3	NCD_4	NCD_5	NCD_6	NCD_7	BL_0	BL_1	...	AS(NA)_6	AS(NA)_7	AS(NAC)_0	AS(NAC)_1	AS(NAC)_
0	0	0	0	0	0	0	0	0	0.00	0.00	...	0.00	0.00	0.00	0.00	0.0
1	0	0	0	0	0	0	0	0	0.00	0.00	...	0.00	0.00	0.00	0.00	0.0
2	0	0	0	0	0	0	0	0	0.00	0.00	...	0.00	0.00	0.00	0.00	0.0
3	0	0	0	0	0	0	0	0	0.00	0.00	...	0.00	0.00	0.00	0.00	0.0
4	0	0	0	0	0	0	0	0	0.00	0.00	...	0.00	0.00	0.00	0.00	0.0

5 rows × 97 columns

```
time: 15 ms (started: 2022-12-26 03:34:58 +05:00)
```

In [202…

```
hardware_classes = hardware_df[['AS(NAC)_7']]
hardware_df.drop(columns = ['AS(NAC)_7'], inplace = True)  #Dropping unnecessary columns

#Scaling and One-hot encoding
Scaler = MinMaxScaler()
hardware_df = pd.get_dummies(hardware_df)
hardware_df = pd.DataFrame(Scaler.fit_transform(hardware_df), columns = hardware_df.columns)
print('Shape of df now is: ', hardware_df.shape)
hardware_df.head()
```

```
Shape of df now is:  (28179, 96)
```

Out[202…

	NCD_0	NCD_1	NCD_2	NCD_3	NCD_4	NCD_5	NCD_6	NCD_7	BL_0	BL_1	...	AS(NA)_5	AS(NA)_6	AS(NA)_7	AS(NAC)_0	AS(NAC)_1
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00

5 rows × 96 columns

```
time: 62 ms (started: 2022-12-26 03:35:10 +05:00)
```

5.3.3 Dimensionality Reduction Techniques with ML

A) Dimensionaliy Reduction Pipeline for Regression Datasets:

In [221…

```
def dimensionality_reduction_regression(X_train, y_train, X_test, y_test, pca_dim = 0.95, ipca_dim= 0.95, svd_dim = 0

    print("Starting DR Pipeline...")

    print("1. Running Lazy Predict without DR")

#Running Lazy Predict without Dimensionality Reduction:
    clf = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric = None)
    simple_models = clf.fit(X_train, X_test, y_train, y_test)[0].sort_index()
    simple_models['dim'] = X_train.shape[1]
    simple_models.drop(columns = [ 'Time Taken'], inplace = True)

    print("Success!")

#Model Performances with PCA:

    #Running PCA:
    print("2. Running PCA")
    pca = PCA(n_components = pca_dim)
    pca.fit(X_train)
    X_train_transformed = pca.transform(X_train)
    X_test_transformed = pca.transform(X_test)
    print("Success!")

    #Running Lazy Predict with PCA
    print("3. Running Lazy Predict on PCA dataset")
    clf = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric = None)
    pcamodels = clf.fit(X_train_transformed, X_test_transformed, y_train, y_test)[0].sort_index()
    pcamodels.drop(columns = [ 'Time Taken'], inplace = True)
```

```
    pcamodels['dims'] = len(pca.components_)
    print("Success!")


#Model Performances with Incremental-PCA:

    #Running Incremental PCA:
    print("4. Running Incremental PCA")
    for i in range(1, X_train.shape[1], 1):
        ipca = IncrementalPCA(n_components = i)
        ipca.fit(X_train)
        X_train_transformed = ipca.transform(X_train)
        X_test_transformed = ipca.transform(X_test)

        if ipca.explained_variance_ratio_.sum() >= ipca_dim:
            print("Success!")
            print("5. Running Lazy Predict on Incremental PCA dataset")
            clf = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric = None)     #Running Lazy Predict after
            ipcamodels = clf.fit(X_train_transformed, X_test_transformed, y_train, y_test)[0].sort_index()
            ipcamodels.drop(columns = [ 'Time Taken'], inplace = True)
            ipcamodels['dims'] = ipca.n_components_
            print("Success!")
            break

#Model Performances with Sparse-PCA:
    print("6. Running Sparse PCA")
    spca = SparsePCA(n_components = 10)
    spca.fit(X_train)
    X_train_transformed = spca.transform(X_train)
    X_test_transformed = spca.transform(X_test)
    print("Success!")

    #Running Lazy Predict Sparse PCA:
    print("7. Running Lazy Predict on Sparse PCA dataset")
    clf = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric = None)     #Running Lazy Predict after SVD
    spcamodels = clf.fit(X_train_transformed, X_test_transformed, y_train, y_test)[0].sort_index()
    spcamodels.drop(columns = [ 'Time Taken'], inplace = True)
    spcamodels['dims'] = spca.n_components_
    print("Success!")


#Model Performances with SVD:

    #Running SVD:
    print("8. Running SVD")
    for i in range(1, X_train.shape[1], 1):
        svd = TruncatedSVD(n_components = i)
        svd.fit(X_train)
        X_train_transformed = svd.transform(X_train)
        X_test_transformed = svd.transform(X_test)

        if svd.explained_variance_ratio_.sum() >= svd_dim or i>=X_train.shape[1]-1:
            print("Success!")
            print("9. Running Lazy Predict on SVD dataset")
            clf = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric = None)     #Running Lazy Predict after
            svdmodels = clf.fit(X_train_transformed, X_test_transformed, y_train, y_test)[0].sort_index()
            svdmodels.drop(columns = [ 'Time Taken'], inplace = True)
            svdmodels['dims'] = len(svd.components_)
            print("Success!")
            break



#Compiling Model Results:
    print("Compiling Model Results")
    models_results = pd.concat([simple_models,
                                pcamodels,
                                ipcamodels,
                                spcamodels,
                                svdmodels], axis = 1, keys =['Without DR', 'PCA ', 'Incremental-PCA', 'Sparse-PCA', 'S

    print("Pipeline run Successful")

    return models_results
```

```
time: 0 ns (started: 2022-12-26 03:51:55 +05:00)
```

B) Applying Pipeline to Regression Datasets

1. Power Consumption Dataset

In [226…

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(power_df, power_classes, test_size=0.25, random_state =43)
```

```
time: 0 ns (started: 2022-12-26 03:57:08 +05:00)
```

In [227…

```
#Pipeline run
models_results = dimensionality_reduction_regression(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = 0.95

#keeping only the desired algorithms
results = models_results.T[['AdaBoostRegressor', 'DecisionTreeRegressor',
       'ElasticNetCV', 'GradientBoostingRegressor', 'KNeighborsRegressor', 'XGBRegressor',
       'RandomForestRegressor', 'SVR']].T

#Exporting results to excel
results.to_excel('Results/Regression/Power_cons.xlsx', sheet_name = 'Power Consumption Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████| 42/42 [1:12:20<00:00, 103.35s/i
t]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [1:07:17<00:00, 96.12s/i
t]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [1:07:15<00:00, 96.08s/i
t]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [1:02:32<00:00, 89.35s/i
t]
Success!
8. Running SVD
Success!
9. Running Lazy Predict on SVD dataset
100%|██████████████████████████████████████████████████| 42/42 [1:12:01<00:00, 102.89s/i
t]
Success!
Compiling Model Results
Pipeline run Successful
time: 5h 41min 27s (started: 2022-12-26 03:57:09 +05:00)
```

Dataset Results:

In [337…

```
#Results
results = pd.read_excel('Results/Regression/Power_cons.xlsx', header=[0, 1], index_col=0 )
results
```

Out[337…

	Without DR				PCA				Incremental-PCA					
	Adjusted R-Squared	**R-Squared**	**RMSE**	**dim**	**Adjusted R-Squared**	**R-Squared**	**RMSE**	**dims**	**Adjusted R-Squared**	**R-Squared**	**RMSE**	**dims**	**Adjusted R-Squared**	**S**
Model														
AdaBoostRegressor	0.90	0.90	5.42	4	0.88	0.88	5.91	3	0.87	0.87	6.14	3	0.90	
DecisionTreeRegressor	0.93	0.93	4.50	4	0.87	0.87	6.20	3	0.86	0.86	6.23	3	0.93	
ElasticNetCV	0.92	0.92	4.69	4	0.88	0.88	5.75	3	0.88	0.88	5.75	3	0.92	
GradientBoostingRegressor	0.94	0.94	4.03	4	0.92	0.92	4.87	3	0.92	0.92	4.89	3	0.94	
KNeighborsRegressor	0.95	0.95	3.87	4	0.92	0.92	4.71	3	0.92	0.92	4.71	3	0.95	
XGBRegressor	0.96	0.96	3.24	4	0.92	0.92	4.74	3	0.92	0.92	4.76	3	0.96	
RandomForestRegressor	0.96	0.96	3.45	4	0.93	0.93	4.52	3	0.93	0.93	4.52	3	0.96	
SVR	0.94	0.94	4.25	4	0.91	0.91	4.99	3	0.91	0.91	4.99	3	0.94	

```
time: 16 ms (started: 2022-12-29 01:24:35 +05:00)
```

Analysis:

For our analysis, I will be using adjusted R2 to make comparison since number of features are changing for each model:

1. When no DR technique is applied, the best adjusted R2 score achieved was 0.96.

2. After applying PCA, the R2 score slightly decreased to 0.93 with a reduction in features to 3 from 4.
3. When applying other PCA variants, Incremental PCA behaved like PCA, but sparse PCA captured all the variance however it increased the dimensionality rather than decreasing it.
4. When SVD was tried, the variance captured was 0.93 slightly less than 0.96 with 1 feature reduction so it performed at par with PCA.

Only 1 feature was reduced in these DR techniques with little compromise to variance capture, PCA and SVD performed at par with each other!

2. Energy Efficiency Dataset

In [228…

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(energy_df, energy_classes, test_size=0.25, random_state =43)
```

```
time: 0 ns (started: 2022-12-26 09:38:36 +05:00)
```

In [229…

```
#Pipeline run
models_results = dimensionality_reduction_regression(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = 0.95

#keeping only the desired algorithms
results = models_results.T[['AdaBoostRegressor', 'DecisionTreeRegressor',
       'ElasticNetCV', 'GradientBoostingRegressor', 'KNeighborsRegressor', 'XGBRegressor',
       'RandomForestRegressor', 'SVR']].T

#Exporting results to excel
results.to_excel('Results/Regression/Energy.xlsx', sheet_name = 'Energy Efficiency Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████| 42/42 [00:02<00:00, 14.14it/
s]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [00:02<00:00, 14.01it/
s]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [00:02<00:00, 14.03it/
s]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [00:03<00:00, 13.78it/
s]
Success!
8. Running SVD
Success!
9. Running Lazy Predict on SVD dataset
100%|██████████████████████████████████████████████████| 42/42 [00:03<00:00, 13.33it/
s]
Success!
Compiling Model Results
Pipeline run Successful
time: 15.4 s (started: 2022-12-26 09:38:36 +05:00)
```

Dataset Results:

In [338…

```
#Results
results = pd.read_excel('Results/Regression/Energy.xlsx', header=[0, 1], index_col=0 )
results
```

Out[338…

	Without DR				PCA				Incremental-PCA				
	Adjusted R-Squared	R-Squared	RMSE	dim	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared
Model													
AdaBoostRegressor	0.97	0.97	1.84	8	0.95	0.95	2.32	5	0.95	0.95	2.23	5	0.97
DecisionTreeRegressor	1.00	1.00	0.47	8	0.98	0.98	1.41	5	0.98	0.98	1.42	5	1.00

	Without DR				PCA				Incremental-PCA				
	Adjusted R-Squared	R-Squared	RMSE	dim	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared
Model													
ElasticNetCV	0.93	0.93	2.68	8	0.92	0.92	2.88	5	0.92	0.92	2.88	5	0.93
GradientBoostingRegressor	1.00	1.00	0.45	8	0.99	0.99	0.96	5	0.99	0.99	0.93	5	1.00
KNeighborsRegressor	0.96	0.96	2.01	8	0.94	0.94	2.50	5	0.94	0.94	2.50	5	0.96
XGBRegressor	1.00	1.00	0.29	8	1.00	1.00	0.70	5	0.99	0.99	0.72	5	1.00
RandomForestRegressor	1.00	1.00	0.44	8	0.99	0.99	0.76	5	0.99	0.99	0.78	5	1.00
SVR	0.94	0.94	2.51	8	0.91	0.91	2.99	5	0.91	0.91	2.99	5	0.93

```
time: 16 ms (started: 2022-12-29 01:24:58 +05:00)
```

Analysis:

Adjusted R2 score is used to compare the results for the energy efficiency dataset:

1. When no DR technique was applied, the best Adjusted R2 score achieved was 1 which means the model captures all the variance available.
2. When PCA is applied, the features reduced to 5 from 8 but the best adjusted R2 score remained at 1 which shows PCA to be a good technique for this dataset.
3. As other PCA variants are tried, incremental PCA performs at par with PCA, but Sparse PCA increases dimension, so it performs poorly.
4. SVD reduced the dimensions dataset dimensions to 6 from 8 only with almost no compromise on variance capture.

PCA seems to perform well for this dataset as it reduces the maximum dimensions and maintains full variance of the dataset!

3. Aquative Toxicity Dataset

In [260…

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(aquatic_df, aquatic_classes, test_size=0.25, random_state =43)
```

```
time: 0 ns (started: 2022-12-26 15:57:25 +05:00)
```

In [261…

```
#Pipeline run
models_results = dimensionality_reduction_regression(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = 0.95

#keeping only the desired algorithms
results = models_results.T[['AdaBoostRegressor', 'DecisionTreeRegressor',
       'ElasticNetCV', 'GradientBoostingRegressor', 'KNeighborsRegressor', 'XGBRegressor',
       'RandomForestRegressor', 'SVR']].T

#Exporting results to excel
results.to_excel('Results/Regression/aquatic.xlsx', sheet_name = 'Aquatic Toxicity Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|████████████████████████████████████████| 42/42 [00:01<00:00, 21.96it/
s]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|████████████████████████████████████████| 42/42 [00:01<00:00, 22.72it/
s]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|████████████████████████████████████████| 42/42 [00:01<00:00, 22.82it/
s]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|████████████████████████████████████████| 42/42 [00:01<00:00, 21.07it/
s]
Success!
8. Running SVD
Success!
9. Running Lazy Predict on SVD dataset
```

```
100%|██████████████████████████████████████████████████| 42/42 [00:01<00:00, 22.06it/
s]
Success!
Compiling Model Results
Pipeline run Successful
time: 9.72 s (started: 2022-12-26 15:57:45 +05:00)
```

Dataset Results:

In [339…

```
#Results
results = pd.read_excel('Results/Regression/Aquatic.xlsx', header=[0, 1], index_col=0 )
results
```

Out[339…

	Without DR				PCA				Incremental-PCA				
	Adjusted R-Squared	R-Squared	RMSE	dim	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared
Model													
AdaBoostRegressor	0.36	0.39	1.22	7	0.29	0.32	1.30	5	0.35	0.37	1.24	5	0.44
DecisionTreeRegressor	0.26	0.30	1.32	7	-0.10	-0.06	1.61	5	0.03	0.06	1.52	5	0.24
ElasticNetCV	0.39	0.42	1.20	7	0.40	0.42	1.19	5	0.40	0.42	1.19	5	0.37
GradientBoostingRegressor	0.42	0.45	1.17	7	0.34	0.37	1.25	5	0.40	0.42	1.19	5	0.40
KNeighborsRegressor	0.36	0.40	1.22	7	0.41	0.43	1.18	5	0.40	0.43	1.19	5	0.35
XGBRegressor	0.37	0.40	1.21	7	0.26	0.29	1.32	5	0.27	0.30	1.31	5	0.36
RandomForestRegressor	0.42	0.45	1.16	7	0.42	0.44	1.17	5	0.43	0.45	1.16	5	0.42
SVR	0.42	0.45	1.16	7	0.41	0.43	1.18	5	0.41	0.43	1.18	5	0.41

```
time: 15 ms (started: 2022-12-29 01:25:25 +05:00)
```

Analysis:

For the aquatic toxicity dataset, adjusted R2 score is used. The models perform poorly which could be the issue with the dataset itself.

1. The best adjusted R2 score achieved without any DR technique was 0.42.
2. After applying PCA, the number of features reduced from 7 to 5 and adjusted R2 improved to 0.44.
3. When other variants of PCA were tried, incremental PCA performed at par with PCA whereas sparse PCA increased dimensions instead.
4. When SVD is applied, the performance improves further to 0.47 whereas the number of features reduces from 7 to 5.

SVD proves to perform better for this dataset while reducing the number of features and improving adjusted R2 score.

4. Seoul Bikes Dataset

In [232…

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(bikes_df, bikes_classes, test_size=0.25, random_state =43)
```

```
time: 32 ms (started: 2022-12-26 10:24:12 +05:00)
```

In [233…

```
#Pipeline run
models_results = dimensionality_reduction_regression(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = 0.95

#keeping only the desired algorithms
results = models_results.T[['AdaBoostRegressor', 'DecisionTreeRegressor',
       'ElasticNetCV', 'GradientBoostingRegressor', 'KNeighborsRegressor', 'XGBRegressor',
       'RandomForestRegressor', 'SVR']].T

#Exporting results to excel
results.to_excel('Results/Regression/Bikes.xlsx', sheet_name = 'Seoul Bikes Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████| 42/42 [44:09<00:00, 63.08s/i
t]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [44:22<00:00, 63.40s/i
t]
Success!
4. Running Incremental PCA
```

```
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|████████████████████████████████████████| 42/42 [43:55<00:00, 62.75s/i
t]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|████████████████████████████████████████| 42/42 [44:00<00:00, 62.87s/i
t]
Success!
8. Running SVD
Success!
9. Running Lazy Predict on SVD dataset
100%|████████████████████████████████████████| 42/42 [47:39<00:00, 68.09s/i
t]
Success!
Compiling Model Results
Pipeline run Successful
time: 3h 44min 8s (started: 2022-12-26 10:24:12 +05:00)
```

Dataset Results:

In [340…

```
#Results
results = pd.read_excel('Results/Regression/Bikes.xlsx', header=[0, 1], index_col=0 )
results
```

Out[340…

	Without DR				PCA				Incremental-PCA				
	Adjusted R-Squared	R-Squared	RMSE	dim	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared
Model													
AdaBoostRegressor	0.57	0.57	422.93	17	0.36	0.37	515.12	8	0.39	0.40	503.04	8	0.44
DecisionTreeRegressor	0.74	0.75	325.76	17	0.59	0.59	415.19	8	0.59	0.60	411.11	8	0.71
ElasticNetCV	0.52	0.53	445.64	17	0.47	0.47	470.78	8	0.47	0.47	470.77	8	0.47
GradientBoostingRegressor	0.84	0.84	258.42	17	0.71	0.71	350.51	8	0.70	0.70	352.47	8	0.81
KNeighborsRegressor	0.78	0.78	299.98	17	0.76	0.76	316.49	8	0.76	0.76	316.98	8	0.73
XGBRegressor	0.87	0.88	228.23	17	0.79	0.79	299.02	8	0.78	0.78	300.94	8	0.85
RandomForestRegressor	0.87	0.87	230.96	17	0.80	0.81	285.21	8	0.81	0.81	284.90	8	0.85
SVR	0.33	0.34	525.78	17	0.29	0.29	544.90	8	0.29	0.29	544.92	8	0.29

```
time: 16 ms (started: 2022-12-29 01:25:53 +05:00)
```

Analysis:

For the bikes sharing dataset, adjusted R2 score is used to compare the results:

1. The best Adjusted R2 score was achieved to be 0.87 with 17 original features without any DR Technique.
2. After applying PCA, the number of features reduced to 8 from 17 with some compromise on variance capture giving an R2 score of 0.81.
3. For the PCA variants, incremental PCA performs like PCA and sparse PCA performs better as it reduces the number of features from 17 to 10 and still captures the same amount of variance giving an adjusted R2 score of 0.86.
4. SVD performs at par with PCA however the number of dimensions is 1 more than PCA.

Sparse PCA performs well in this dataset as it reduces the number of features without any compromise on variance capture!

5. Red Wine Quality Dataset

In [234…

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(redwine_df, redwine_classes, test_size=0.25, random_state =43)
```

```
time: 0 ns (started: 2022-12-26 14:08:20 +05:00)
```

In [235…

```
#Pipeline run
models_results = dimensionality_reduction_regression(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = 0.95

#keeping only the desired algorithms
results = models_results.T[['AdaBoostRegressor', 'DecisionTreeRegressor',
         'ElasticNetCV', 'GradientBoostingRegressor', 'KNeighborsRegressor', 'XGBRegressor',
         'RandomForestRegressor', 'SVR']].T
```

```
#Exporting results to excel
results.to_excel('Results/Regression/redwine.xlsx', sheet_name = 'Red Wine Quality Dataset')
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|████████████████████████████████████████| 42/42 [00:08<00:00,  5.13it/
s]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|████████████████████████████████████████| 42/42 [00:08<00:00,  5.11it/
s]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|████████████████████████████████████████| 42/42 [00:08<00:00,  5.17it/
s]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|████████████████████████████████████████| 42/42 [00:08<00:00,  5.20it/
s]
Success!
8. Running SVD
Success!
9. Running Lazy Predict on SVD dataset
100%|████████████████████████████████████████| 42/42 [00:08<00:00,  5.12it/
s]
Success!
Compiling Model Results
Pipeline run Successful
time: 41.2 s (started: 2022-12-26 14:08:20 +05:00)
```

Dataset Results:

In [341…

```
#Results
results = pd.read_excel('Results/Regression/redwine.xlsx', header=[0, 1], index_col=0 )
results
```

Out[341…

	Without DR				PCA				Incremental-PCA				
	Adjusted R-Squared	R-Squared	RMSE	dim	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared
Model													
AdaBoostRegressor	0.36	0.37	0.64	11	0.35	0.36	0.64	8	0.34	0.35	0.65	8	0.37
DecisionTreeRegressor	0.01	0.04	0.79	11	-0.02	0.00	0.80	8	-0.04	-0.02	0.81	8	-0.07
ElasticNetCV	0.36	0.37	0.64	11	0.35	0.36	0.64	8	0.35	0.36	0.64	8	0.36
GradientBoostingRegressor	0.41	0.43	0.61	11	0.39	0.40	0.62	8	0.41	0.42	0.61	8	0.42
KNeighborsRegressor	0.30	0.32	0.67	11	0.29	0.30	0.67	8	0.29	0.31	0.67	8	0.32
XGBRegressor	0.41	0.43	0.61	11	0.36	0.37	0.64	8	0.40	0.41	0.62	8	0.41
RandomForestRegressor	0.47	0.49	0.57	11	0.47	0.48	0.58	8	0.48	0.49	0.58	8	0.47
SVR	0.40	0.42	0.61	11	0.40	0.41	0.62	8	0.40	0.41	0.62	8	0.40

```
time: 15 ms (started: 2022-12-29 01:26:26 +05:00)
```

Analysis:

For the Red wine quality dataset, adjusted R2 score is used to compare the results:

1. The best Adjusted R2 score was achieved to be 0.47 with 11 original features without any DR Technique.
2. After applying PCA, the number of features reduced to 8 from 11 with no difference in R2 score proving it to be a particularly good DR technique for this dataset
3. For the PCA variants, both incremental and sparse performed at par with PCA
4. SVD performs reduces number of features to 8 from 11 which is at par with PCA however the adjusted R2 slightly improves but 0.01 difference is not generalizable.

PCA, its variants and SVD all perform well for this dataset!

6. Student Portugese Dataset

In [272…

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(student_df, student_classes, test_size=0.25, random_state =43)
```

```
time: 0 ns (started: 2022-12-26 16:15:29 +05:00)
```

In [273…

```
#Pipeline run
models_results = dimensionality_reduction_regression(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = 0.95

#keeping only the desired algorithms
results = models_results.T[['AdaBoostRegressor', 'DecisionTreeRegressor',
       'ElasticNetCV', 'GradientBoostingRegressor', 'KNeighborsRegressor', 'XGBRegressor',
       'RandomForestRegressor', 'SVR']].T

#Exporting results to excel
results.to_excel('Results/Regression/Student.xlsx', sheet_name = "Student Portugese Dataset")
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|████████████████████████████████████████| 42/42 [00:03<00:00, 13.86it/
s]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|████████████████████████████████████████| 42/42 [00:03<00:00, 13.54it/
s]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|████████████████████████████████████████| 42/42 [00:03<00:00, 13.49it/
s]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|████████████████████████████████████████| 42/42 [00:02<00:00, 19.11it/
s]
Success!
8. Running SVD
Success!
9. Running Lazy Predict on SVD dataset
100%|████████████████████████████████████████| 42/42 [00:03<00:00, 13.20it/
s]
Success!
Compiling Model Results
Pipeline run Successful
time: 15.5 s (started: 2022-12-26 16:15:30 +05:00)
```

Dataset Results:

In [342…

```
#Results
results = pd.read_excel('Results/Regression/Student.xlsx', header=[0, 1], index_col=0 )
results
```

Out[342…

	Without DR				PCA				Incremental-PCA				
	Adjusted R-Squared	R-Squared	RMSE	dim	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared
Model													
AdaBoostRegressor	0.69	0.80	1.47	58	0.04	0.21	2.91	29	-0.02	0.16	3.00	29	-0.04
DecisionTreeRegressor	0.46	0.65	1.93	58	-1.12	-0.74	4.32	29	-0.87	-0.54	4.06	29	-1.46
ElasticNetCV	0.73	0.83	1.36	58	0.09	0.25	2.83	29	0.09	0.25	2.83	29	0.08
GradientBoostingRegressor	0.69	0.80	1.46	58	0.03	0.21	2.92	29	0.03	0.20	2.93	29	-0.14
KNeighborsRegressor	0.18	0.47	2.38	58	0.01	0.19	2.95	29	-0.06	0.13	3.05	29	-0.11
XGBRegressor	0.64	0.77	1.57	58	-0.02	0.16	3.00	29	0.01	0.19	2.95	29	-0.33
RandomForestRegressor	0.70	0.81	1.43	58	0.09	0.25	2.83	29	0.07	0.24	2.86	29	0.00
SVR	0.49	0.67	1.87	58	0.10	0.26	2.82	29	0.08	0.24	2.85	29	0.02

```
time: 15 ms (started: 2022-12-29 01:26:53 +05:00)
```

Analysis:

Adjusted R2 score is used to compare the results for the student performance dataset:

1. When no DR technique was applied, the best Adjusted R2 score achieved was 0.70.
2. When PCA is applied, the features reduced from 58 to 29 but the best the adjusted R2 score drops significantly which means PCA is not a good DR technique for this dataset.
3. The rest of the PCA variants and SVD performs poorly as well! **This dataset does not seem to allow any feature reduction which could be highlighting that there is a remarkably high correlation between some variables with predicted value!**

7. Tom's Hardware Dataset

In [238…

```
#lets split the data into a train test split from the start, test set will be kept separate and will only be used for

X_train, X_test, y_train, y_test = train_test_split(hardware_df, hardware_classes, test_size=0.25, random_state =43)
```

```
time: 94 ms (started: 2022-12-26 14:09:39 +05:00)
```

In [239…

```
#Pipeline run
models_results = dimensionality_reduction_regression(X_train, y_train, X_test, y_test, pca_dim = 0.95, svd_dim = 0.95

#keeping only the desired algorithms
results = models_results.T[['AdaBoostRegressor', 'DecisionTreeRegressor',
       'ElasticNetCV', 'GradientBoostingRegressor', 'KNeighborsRegressor', 'XGBRegressor',
       'RandomForestRegressor', 'SVR']].T

#Exporting results to excel
results.to_excel('Results/Regression/hardware.xlsx', sheet_name = "Tom's Hardware Dataset")
```

```
Starting DR Pipeline...
1. Running Lazy Predict without DR
100%|██████████████████████████████████████████████████| 42/42 [13:19<00:00, 19.02s/i
t]
Success!
2. Running PCA
Success!
3. Running Lazy Predict on PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [10:54<00:00, 15.58s/i
t]
Success!
4. Running Incremental PCA
Success!
5. Running Lazy Predict on Incremental PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [11:01<00:00, 15.76s/i
t]
Success!
6. Running Sparse PCA
Success!
7. Running Lazy Predict on Sparse PCA dataset
100%|██████████████████████████████████████████████████| 42/42 [10:26<00:00, 14.91s/i
t]
Success!
8. Running SVD
Success!
9. Running Lazy Predict on SVD dataset
100%|██████████████████████████████████████████████████| 42/42 [10:52<00:00, 15.54s/i
t]
Success!
Compiling Model Results
Pipeline run Successful
time: 56min 41s (started: 2022-12-26 14:09:39 +05:00)
```

Dataset Results:

In [343…

```
#Results
results = pd.read_excel('Results/Regression/hardware.xlsx', header=[0, 1], index_col=0 )
results
```

Out[343…

	Without DR				PCA				Incremental-PCA				
	Adjusted R-Squared	R-Squared	RMSE	dim	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared	R-Squared	RMSE	dims	Adju Squ
Model													
AdaBoostRegressor	0.88	0.88	4458.21	96	-1.85	-1.85	21948.76	18	-1.61	-1.61	21005.85	18	
DecisionTreeRegressor	0.92	0.92	3697.34	96	0.79	0.79	5893.47	18	0.82	0.82	5523.79	18	

	Without DR				PCA				Incremental-PCA				
	Adjusted R-Squared	R-Squared	RMSE	dim	Adjusted R-Squared	R-Squared	RMSE	dims	Adjusted R-Squared	R-Squared	RMSE	dims	Adju Squ
Model													
ElasticNetCV	0.83	0.83	5304.48	96	0.15	0.16	11946.31	18	0.15	0.16	11946.38	18	
GradientBoostingRegressor	0.97	0.97	2354.97	96	0.90	0.90	4118.33	18	0.90	0.90	4021.61	18	
KNeighborsRegressor	0.95	0.95	3006.69	96	0.88	0.88	4498.94	18	0.88	0.88	4495.64	18	
XGBRegressor	0.97	0.97	2290.19	96	0.90	0.90	4063.83	18	0.90	0.90	4057.10	18	
RandomForestRegressor	0.97	0.97	2297.70	96	0.90	0.90	4143.08	18	0.90	0.90	4199.52	18	
SVR	-0.03	-0.02	13124.97	96	-0.03	-0.03	13216.94	18	-0.03	-0.03	13216.98	18	

```
time: 15 ms (started: 2022-12-29 01:27:20 +05:00)
```

Analysis:

Adjusted R2 score is used to compare the results for the Tom's Hardware dataset:

1. When no DR technique was applied, the best Adjusted R2 score achieved was 0.97 which means the model captures almost all the variance available.
2. When PCA is applied, the features reduced from 96 to 18 but with a small drop in adjusted R2 score of 0.06 which leads to an R2 score of 0.90.
3. As other PCA variants are tried, incremental PCA performs at par with PCA but sparse reduces dimensions to 10 with some more compromise on adjusted R2 value leading to R2 score of 0.87
4. SVD performs at par with PCA as it leads to an adjusted R2 score of 0.9 and gives a feature set of 18 dimensions.

PCA and SVD both perform well as they reduce the number of features from 96 to 18 and capture all variance!

6. Critical Analysis

All the datasets do not compromise a lot on variance capture when their dimensionality is reduced using any of the techniques tried above which proves that these techniques are especially useful and should be implemented before diving deep into machine learning. Spending some time on reducing dimensions would help in the long run since model development becomes easier and less time-consuming when dataset features are reduced. Furthermore, LDA is only applicable on classification datasets and there are other varying factors which does not allow generalizability amongst all datasets however, it can be concurred that PCA works well for all datasets as it helps reduce dimensions more significantly and does not compromise a lot on variance capture which is the major goal of PCA itself. The major reason that these techniques work so well is because tabular data can be explained via linear mappings. Since PCA, LDA and SVD all are linear transformers, they can capture the hidden trends in the data well. However, same cannot be said for textual or image data where there are non-linear patterns.

6.1 Classification Datasets

Most of the techniques performed well but Linear discriminant analysis stood out for every dataset. LDA significantly reduced dimensions without compromising on variance capture making it an extremely useful technique. The major reason LDA can outperform the other techniques is because it is a supervised one which could be considered its drawback as well. However, since we are focused on classification, we would require a labeled dataset and LDA uses the labels along with the dataset to increase separability between classes. PCA only focuses on the linear mappings between the predictor variable whereas LDA focuses on linear mapping between the entire dataset and predicted variable as well. In addition, LDA's focus on class separability becomes the major factor that helps in improving classification since LDA focuses on capturing distinct information between the classes. However, LDA forcefully reduces dimensions to less than number of classes which may cause losing essential information if the entire dataset is relevant so we must resort to PCA or SVD if there is a major performance drop with LDA.

6.2 Regression Datasets

For regression datasets, LDA was not applicable as it requires a classification dataset. PCA worked well for these as it was able to capture most of the variance with significant feature reduction. Singular value decomposition performed like PCA as well where in some cases it was able to surpass PCA however PCA still remained the winner technique for regression datasets. The reason PCA works so well is because it can remove multicollinearity between variables and map the dataset on independent dimensions that capture the highest variance. Furthermore, because regression has a continuous output, it is more prone to having noise than classification datasets so amongst the other techniques PCA can handle noise better.

7. Conclusion

Overall, DR techniques must become a standard pre-processing step for high dimensional datasets to avoid unnecessary prolonged computation time and make machine learning simpler.