**Dimension Reduction – PCA & SVD**

**Instructions:**

Please share your answers wherever applicable in line with the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: Vishvash C Batch ID:** 23012024

**Topic: Dimension Reduction – PCA & SVD**

**Guidelines:**

**1. An assignment submission is considered complete only when correct and executable code(s) is submitted along with the documentation, explaining the method and results. Failing to submit either of those will be considered an invalid submission and will not be considered a correct submission.**

**2. Ensure that you submit your assignments correctly and in full. Resubmission is not allowed.**

**3. Post the submission you can evaluate your work by referring to the keys provided. (Will be available only post the submission).**

**Hints:**

**1. Business Problem**

* 1. **What is the business objective?**
  2. **What are the constraints?**
  3. **Define success criteria**

**2. Work on each feature of the dataset to create a data dictionary as displayed in the below image:**

Table

Description automatically generated

**3. Exploratory Data Analysis (EDA):**

**3.1. Univariate analysis.**

**3.2. Bivariate analysis.**

**4. Data Pre-processing**

**4.1 Data Cleaning, Feature Engineering, etc.**

**5.** **Multivariate Analysis**

**5.1 Build the model on the scaled data (try multiple options).**

**5.2 Perform the clustering and analyze the clusters.**

**5.3 Validate the clusters (try with the different numbers of clusters), label the clusters, and derive insights (compare the results from multiple approaches).**

**6. Use the clustered data and perform feature extraction using PCA and SVD. Compare the results.**

**7. Write about the benefits/impact of the solution - in what way does the business (client) benefit from the solution provided?**

**8. Deploy the best model using Python Flask on the local machine.**

**Problem Statements:**

The average retention rate in the insurance industry is 84%, with the top-performing agencies in the 93% - 95% range. Retaining customers is all about the long-term relationship you build. Offering a discount on the client’s current policy will ensure he/she buys a new product or renews the current policy. Studying clients' purchasing behaviour to determine which products they're most likely to buy is essential.

The insurance company wants to analyze their customer’s behaviour to strategies offers to increase customer loyalty.

**CRISP-ML(Q) process model describes six phases:**

1. Business and Data Understanding

2. Data Preparation

3. Model Building

4. Model Evaluation

5. Deployment

6. Monitoring and Maintenance

**Objective**: Maximize the Sales

**Constraints**: Minimize the Customer Retention

**Success Criteria:**

Business Success Criteria: Increase the Sales by 10% to 12% by targeting cross-selling opportunities on current customers.

ML Success Criteria: NA

Economic Success Criteria: The insurance company will see an increase in revenues by at least 8%

Data: Refer to the Autoinsurance.csv dataset.A picture containing chart

Description automatically generated

**Questions to Trigger Your thoughts:**

Q1. Which libraries are used in PCA to find the optimal number of PCA components?

The scikit-learn library is commonly used in PCA to find the optimal number of PCA components. This can be done using techniques like Scree Plot, Cumulative Explained Variance Plot, or Automated Methods like Grid Search.

Q2. Principal Component Analysis (PCA) is a \_\_\_\_\_\_\_\_\_\_ technique in Data Mining?

Principal Component Analysis (PCA) is a dimensionality reduction technique in Data Mining.

Q3. What is the importance of using PCA before clustering?

PCA helps in reducing the dimensionality of the data by transforming it into a lower-dimensional space while preserving most of its variance. This can lead to improved clustering results by reducing noise and computational complexity.

Q4. Can we perform PCA on categorical features?

No, PCA is typically applied to numerical features. Categorical features need to be preprocessed using techniques like one-hot encoding before PCA.

Q5. Why is it important to create pipelines?

Pipelines help in streamlining the workflow by combining multiple preprocessing and modeling steps into a single object. This ensures consistency, reproducibility, and ease of deployment.

Q6. Which libraries can we use to save or dump pipelines?

Libraries like joblib and pickle in Python can be used to save or dump pipelines.

Q7. Why is it important to standardize the data in PCA?

Standardizing the data ensures that all features have the same scale, which is crucial for PCA since it is sensitive to the scale of the features. Standardization helps in giving equal importance to all features during PCA.

Q8. How can you obtain the principal components and the eigenvalues from Scikit-Learn PCA?

In Scikit-Learn, you can obtain the principal components using the components\_ attribute and the eigenvalues using the explained\_variance\_ attribute of the PCA object.

Q9. What is sklearn.pipeline extension used for?

The sklearn.pipeline extension is used for constructing and managing machine learning pipelines in scikit-learn. It helps in chaining together multiple data preprocessing and modeling steps into a single object.

Q10. Why do we use the filterwarnings function? What library does it belong to, and what are the uses of the library?

The filterwarnings function belongs to the warnings module in Python. It is used to control which warnings should be issued and how they should be displayed. This can help in managing warning messages during code execution.

Q11. What is the extension for the sklearn library to import TruncatedSVD?

The extension for importing TruncatedSVD from the sklearn library is from sklearn.decomposition import TruncatedSVD.

Q12. How to read only the first 30 data rows?

You can read only the first 30 data rows using slicing in Python. For example, if you're using pandas DataFrame, you can use df.head(30) to read the first 30 rows.

Q13. What are the common functions used from the joblib library? Why do we use this library?

Common functions used from the joblib library include dump, load, and Parallel. joblib is used for efficient storage of Python objects to disk and for parallel computing, especially in machine learning tasks.

Q14. How to drop columns in location [5]?

To drop columns at location 5, you can use indexing in Python. For example, if you're working with a pandas DataFrame, you can use df.drop(df.columns[5], axis=1) to drop the column at location 5.

Q15. How to set the timeframe as an index?

To set the timeframe as an index in pandas DataFrame, you can use the set\_index function. For example, if 'timeframe' is the name of your timeframe column, you can use df.set\_index('timeframe', inplace=True).

Q16. How to check what imputation is better for replacing nan/infinity values?

You can check the effectiveness of different imputation techniques for replacing NaN/Infinity values by evaluating their impact on the model performance using techniques like cross-validation and comparing different imputation strategies.

Q17. What does figsize(x, y) define in plotting?

figsize(x, y) defines the dimensions (width and height) of the figure in a matplotlib plot. It specifies the size of the figure in inches.

Q18. Can we define the type of plot inside a plot() function?

Yes, you can define the type of plot inside the plot() function by specifying the kind of plot using parameters like kind or by using different plot functions provided by the plotting library.

Q19. How is SVD different from PCA?

PCA is primarily used for dimensionality reduction by finding a new set of orthogonal axes (principal components) that capture the maximum variance in the data. These principal components are linear combinations of the original features.

On the other hand, SVD is a matrix factorization technique that decomposes a matrix into three matrices: U, Σ, and V^T. While SVD can also be used for dimensionality reduction, it's more general-purpose and has applications beyond PCA. SVD can be applied to any matrix, not just covariance matrices, making it more versatile.

Q20. What are n\_components in SVD?

In Singular Value Decomposition (SVD), n\_components refers to the number of singular values and vectors to keep in the truncated representation of the decomposed matrices.

Q21. What does the fit function do? What does the transform function do?

In scikit-learn, the fit() function is used to com­pute the parameters needed to perform a specific operation (e.g., training a model), while the transform() function applies these parameters to transform the data based on the computed parameters.



**PCA**

# -\*- coding: utf-8 -\*-

"""

Created on Sun Mar 24 18:02:44 2024

@author: Lenovo

"""

'''

Problem Statements:

The average retention rate in the insurance industry is 84%, with the top-performing agencies in the 93% - 95% range.

Retaining customers is all about the long-term relationship you build.

Offering a discount on the client’s current policy will ensure he/she buys a new product or renews the current policy.

Studying clients' purchasing behaviour to determine which products they're most likely to buy is essential.

The insurance company wants to analyze their customer’s behaviour to strategies offers to increase customer loyalty.

CRISP-ML(Q) process model describes six phases:

1. Business and Data Understanding

2. Data Preparation

3. Model Building

4. Model Evaluation

5. Deployment

6. Monitoring and Maintenance

Objective: Maximize the Sales

Constraints: Minimize the Customer Retention

Success Criteria:

Business Success Criteria: Increase the Sales by 10% to 12% by targeting cross-selling opportunities on current customers.

ML Success Criteria: NA

Economic Success Criteria: The insurance company will see an increase in revenues by at least 8%

'''

# #### Install the required packages if not available

# !pip install feature\_engine

# !pip install dtale

# \*\*Importing required packages\*\*

import numpy as np

import pandas as pd

import sweetviz

import matplotlib.pyplot as plt

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import make\_pipeline

from sklearn.decomposition import PCA

from kneed import KneeLocator

from sqlalchemy import create\_engine, text

user = 'root' # user name

pw = '1234' # password

db = 'insur\_db' # database

# creating engine to connect database

engine = create\_engine(f"mysql+pymysql://{user}:{pw}@localhost/{db}")

# \*\*Import the data\*\*

insur = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/Dimension Reduction/Assignments/key/AutoInsurance (1).csv")

insur

# dumping data into database

# name should be in lower case

insur.to\_sql('insur\_clustering', con = engine, if\_exists = 'replace', chunksize = 1000, index = False)

# loading data from database

sql = text('select \* from insur\_clustering')

df = pd.read\_sql\_query(sql, con = engine.connect())

print(df)

df.info()

# # EXPLORATORY DATA ANALYSIS (EDA) / DESCRIPTIVE STATISTICS

# \*\*\*Descriptive Statistics and Data Distribution Function\*\*\*

res = df.describe()

# Handle duplicates

df.Customer.duplicated().sum()

# Filter the numerical columns

df1 = df.select\_dtypes(exclude=['object'])

# AutoEDA

# Automated Libraries

# import sweetviz

my\_report = sweetviz.analyze([df1, "df1"])

my\_report.show\_html('Report.html')

# Data Preprocessing

# Checking Null Values

df1.isnull().sum()

# PCA can be implemented only on Numeric features

df1.info()

numeric\_features = df1.select\_dtypes(exclude = ['object']).columns

numeric\_features

# Define the Pipeline steps

# Define PCA model

pca = PCA(n\_components = 8)

# Make Pipeline

# \*\*By using mean imputation, null values can be imputed\*\*

# \*\*Data has to be standardized to address the scale difference\*\*

num\_pipeline = make\_pipeline(SimpleImputer(strategy = 'mean'), StandardScaler(), pca)

num\_pipeline

# Pass the raw data through pipeline

processed = num\_pipeline.fit(df1[numeric\_features])

processed

# Save the End to End PCA pipeline with Imputation and Standardization

import joblib

joblib.dump(processed, 'PCA\_DimRed')

import os

os.getcwd()

# Import the pipeline

model = joblib.load("PCA\_DimRed")

model

# ## Apply the saved model on to the Dataset to extract PCA values

pca\_res = pd.DataFrame(model.transform(df1[numeric\_features]))

pca\_res

# PCA weights

model['pca'].components\_

# Take a closer look at the components

components = pd.DataFrame(model['pca'].components\_, columns = numeric\_features).T

components.columns = ['pc0', 'pc1', 'pc2', 'pc3', 'pc4', 'pc5', 'pc6', 'pc7']

components

print(model['pca'].explained\_variance\_ratio\_)

var1 = np.cumsum(model['pca'].explained\_variance\_ratio\_)

print(var1)

# Variance plot for PCA components obtained

plt.plot(var1, color = "red")

# KneeLocator

# Refer the link to understand the parameters used: https://kneed.readthedocs.io/en/stable/parameters.html

# from kneed import KneeLocator

kl = KneeLocator(range(len(var1)), var1, curve = 'convex', direction = "increasing")

# The line is pretty linear hence Kneelocator is not able to detect the knee/elbow appropriately

kl.elbow

# plt.style.use("seaborn")

plt.plot(range(len(var1)), var1)

plt.xticks(range(len(var1)))

plt.ylabel("variance")

plt.axvline(x = kl.elbow, color = 'r', label = 'axvline - full height', ls = '--')

plt.show()

# The line is pretty linear hence Kneelocator is not able to detect the knee/elbow appropriately

# PCA for Feature Extraction

plt.plot(range(len(var1)), var1)

plt.xticks(range(len(var1)))

plt.ylabel("variance")

plt.axvline(x = 3, color = 'r', label = 'axvline - full height', ls = '--')

plt.show()

# Final dataset with manageable number of columns (Feature Extraction)

final = pd.concat([df.Customer, pca\_res.iloc[:, 0:4]], axis = 1)

final.columns = ['Customer', 'pc0', 'pc1', 'pc2', 'pc3']

final

# Scatter diagram

ax = final.plot(x = 'pc0', y = 'pc1', kind = 'scatter', figsize = (12, 8))

final[['pc0', 'pc1', 'Customer']].apply(lambda x: ax.text(\*x), axis = 1)

**Output:**

pca\_res

Out[99]:

0 1 2 ... 5 6 7

0 -0.988514 -0.445305 1.738864 ... 0.068370 -0.572824 -0.428192

1 1.760618 -1.579777 0.391470 ... -0.319258 -0.491658 -1.180024

2 0.767736 0.560896 0.406998 ... -0.047913 0.067237 -0.235184

3 0.750825 -0.946134 0.169339 ... 0.696704 0.083742 0.417082

4 -1.355272 0.007026 -0.427308 ... -0.293771 -0.076998 0.343988

... ... ... ... ... ... ...

9129 -0.160202 2.181523 -0.395019 ... 1.396783 1.626760 -0.512022

components

Out[104]:

pc0 pc1 ... pc6 pc7

Customer Lifetime Value 0.415830 0.469770 ... 0.751238 -0.135124

Income -0.242604 0.782291 ... -0.440413 -0.310546

Monthly Premium Auto 0.615247 0.224945 ... -0.367509 0.653712

Months Since Last Claim 0.019734 -0.183018 ... -0.052956 -0.007627

Months Since Policy Inception 0.019469 0.128531 ... -0.007441 -0.014015

Number of Open Complaints -0.036615 -0.104806 ... 0.069760 -0.005442

Number of Policies 0.002970 -0.016142 ... -0.081174 0.008127

Total Claim Amount 0.622561 -0.235534 ... -0.303809 -0.676471

print(model['pca'].explained\_variance\_ratio\_)

[0.23943177 0.14027837 0.13075081 0.12490888 0.12444049 0.11955291

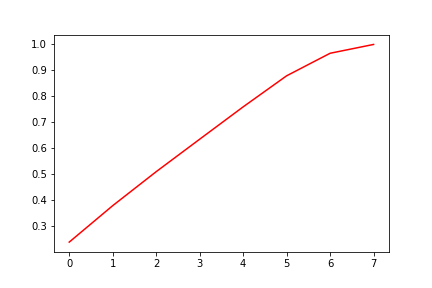
0.08670387 0.03393292]

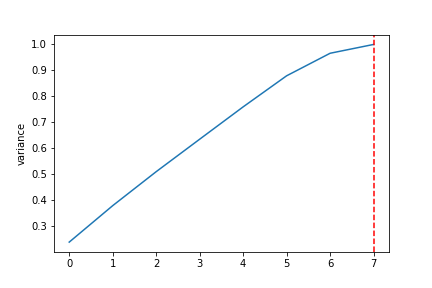
var1 = np.cumsum(model['pca'].explained\_variance\_ratio\_)

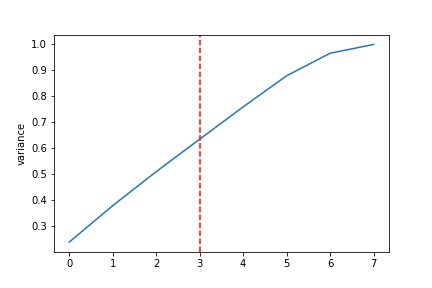
print(var1)

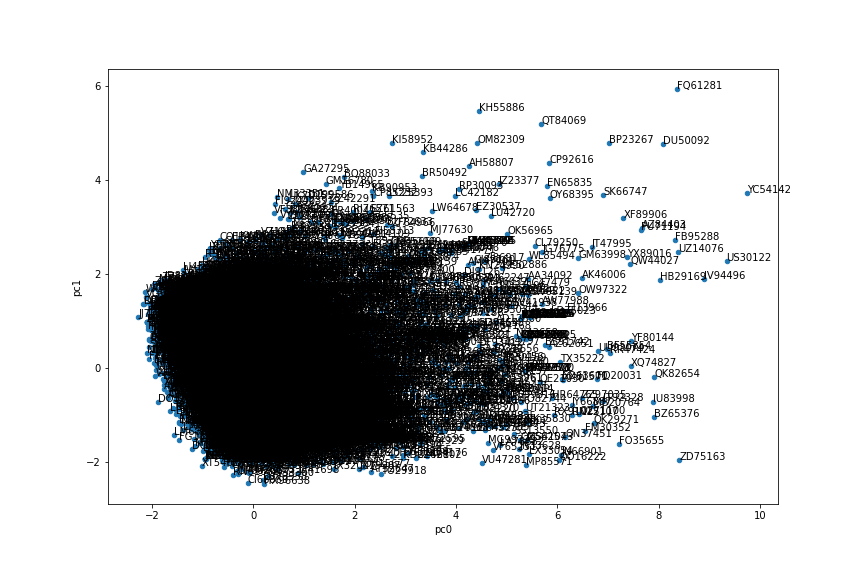
[0.23943177 0.37971013 0.51046095 0.63536982 0.75981031 0.87936322

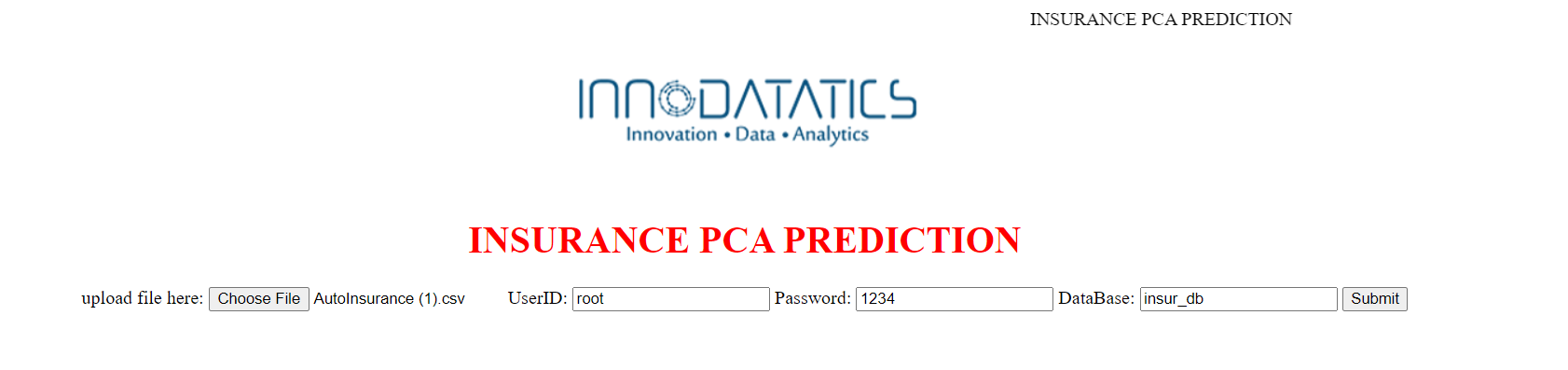
0.96606708 1.

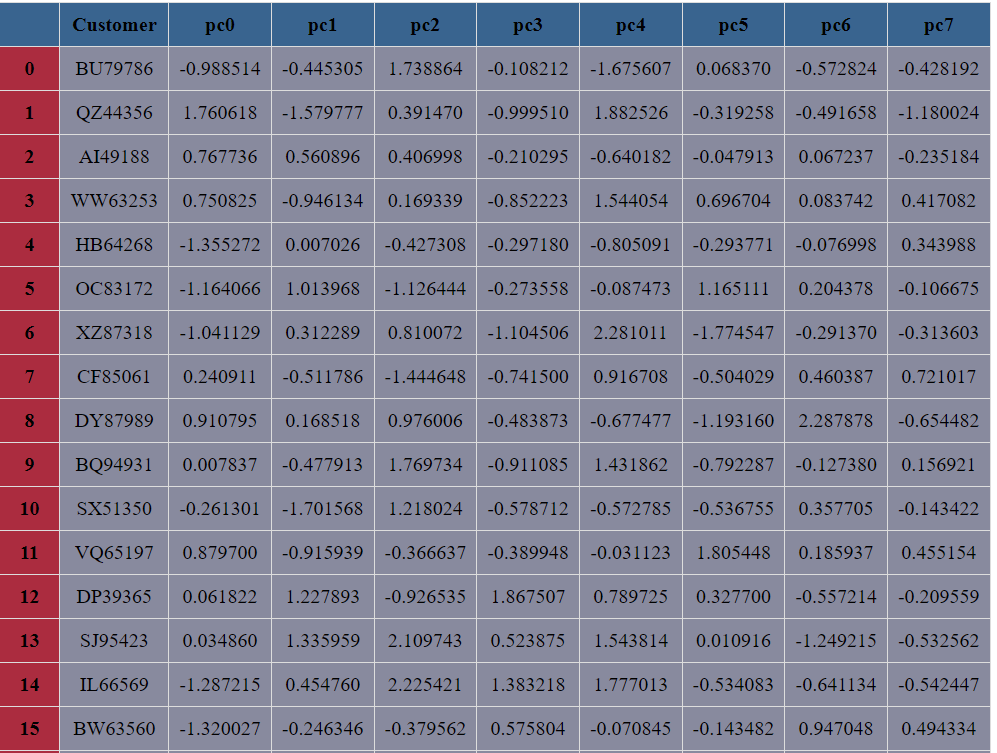








**Deployment of PCA using Flask**



**SVD**

# -\*- coding: utf-8 -\*-

"""

Created on Sun Mar 24 19:02:06 2024

@author: Lenovo

"""

# #### Install the required packages if not available

# !pip install feature\_engine

# !pip install dtale

# \*\*Importing required packages\*\*

import numpy as np

import pandas as pd

import sweetviz

import matplotlib.pyplot as plt

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import make\_pipeline

from sklearn.decomposition import TruncatedSVD

from kneed import KneeLocator

from sqlalchemy import create\_engine, text

user = 'root' # user name

pw = '1234' # password

db = 'insur\_db' # database

# creating engine to connect database

engine = create\_engine(f"mysql+pymysql://{user}:{pw}@localhost/{db}")

# \*\*Import the data\*\*

insur = pd.read\_csv(r"C:/Users/Lenovo/Downloads/Study material/Data Science/Dimension Reduction/Assignments/key/AutoInsurance (1).csv")

insur

# dumping data into database

# name should be in lower case

insur.to\_sql('insur\_clustering', con = engine, if\_exists = 'replace', chunksize = 1000, index = False)

# loading data from database

sql = text('select \* from insur\_clustering')

df = pd.read\_sql\_query(sql, con = engine.connect())

print(df)

df.info()

# # EXPLORATORY DATA ANALYSIS (EDA) / DESCRIPTIVE STATISTICS

# \*\*\*Descriptive Statistics and Data Distribution Function\*\*\*

res = df.describe()

# Handle duplicates

df.Customer.duplicated().sum()

# Filter the numerical columns

df1 = df.select\_dtypes(exclude=['object'])

# AutoEDA

# Automated Libraries

# import sweetviz

my\_report = sweetviz.analyze([df1, "df1"])

my\_report.show\_html('Report.html')

# Data Preprocessing

# Checking Null Values

df1.isnull().sum()

# SVD can be implemented only on Numeric features

df1.info()

numeric\_features = df1.select\_dtypes(exclude = ['object']).columns

numeric\_features

# Define the Pipeline steps

# Define svd model

svd = TruncatedSVD(n\_components = 5)

# Make Pipeline

# \*\*By using mean imputation, null values can be imputed\*\*

# \*\*Data has to be standardized to address the scale difference\*\*

num\_pipeline = make\_pipeline(SimpleImputer(strategy = 'mean'), StandardScaler(), svd)

num\_pipeline

# Pass the raw data through pipeline

processed = num\_pipeline.fit(df1[numeric\_features])

processed

# Save the End to End svd pipeline with Imputation and Standardization

import joblib

joblib.dump(processed, 'SVD\_DimRed')

import os

os.getcwd()

# Import the pipeline

model = joblib.load("SVD\_DimRed")

model

# ## Apply the saved model on to the Dataset to extract svd values

svd\_res = pd.DataFrame(model.transform(df1[numeric\_features]))

svd\_res

# svd weights

svd.components\_

# Take a closer look at the components

components = pd.DataFrame(svd.components\_, columns = numeric\_features).T

components.columns = ['svd0', 'svd1', 'svd2', 'svd3', 'svd4']

components

print(svd.explained\_variance\_ratio\_)

var1 = np.cumsum(svd.explained\_variance\_ratio\_)

print(var1)

# Variance plot for svd components obtained

plt.plot(var1, color = "red")

# KneeLocator

# Refer the link to understand the parameters used: https://kneed.readthedocs.io/en/stable/parameters.html

# from kneed import KneeLocator

kl = KneeLocator(range(len(var1)), var1, curve = 'convex', direction = "increasing")

# The line is pretty linear hence Kneelocator is not able to detect the knee/elbow appropriately

kl.elbow

# plt.style.use("seaborn")

plt.plot(range(len(var1)), var1)

plt.xticks(range(len(var1)))

plt.ylabel("variance")

plt.axvline(x = kl.elbow, color = 'r', label = 'axvline - full height', ls = '--')

plt.show()

# The line is pretty linear hence Kneelocator is not able to detect the knee/elbow appropriately

# svd for Feature Extraction

plt.plot(range(len(var1)), var1)

plt.xticks(range(len(var1)))

plt.ylabel("variance")

plt.axvline(x = 3, color = 'r', label = 'axvline - full height', ls = '--')

plt.show()

# Final dataset with manageable number of columns (Feature Extraction)

final = pd.concat([df.Customer, svd\_res.iloc[:, 0:4]], axis = 1)

final.columns = ['Customer', 'svd0', 'svd1', 'svd2', 'svd3']

final

# Scatter diagram

ax = final.plot(x = 'svd1', y = 'svd2', kind = 'scatter', figsize = (12, 8))

final[['svd1', 'svd2', 'Customer']].apply(lambda x: ax.text(\*x), axis = 1)

**Output:**

num\_pipeline

Out[131]:

Pipeline(steps=[('simpleimputer', SimpleImputer()),

('standardscaler', StandardScaler()),

('truncatedsvd', TruncatedSVD(n\_components=5))])

svd\_res = pd.DataFrame(model.transform(df1[numeric\_features]))

svd\_res

Out[146]:

0 1 2 3 4

0 -0.988514 -0.445305 1.738864 -0.108212 -1.675607

1 1.760618 -1.579777 0.391470 -0.999510 1.882526

2 0.767736 0.560896 0.406998 -0.210295 -0.640182

3 0.750825 -0.946134 0.169339 -0.852223 1.544054

4 -1.355272 0.007026 -0.427308 -0.297180 -0.805091

... ... ... ... ...

9129 -0.160202 2.181523 -0.395019 -0.182010 -0.215350

9130 -0.543906 -0.812424 -0.078823 -0.349737 -0.951643

9131 0.801777 -1.536623 -0.477437 2.647552 0.454271

9132 0.718288 -1.134879 2.092450 -0.324251 -0.933533

9133 -0.435340 -0.921750 -2.338228 -0.491840 -0.111991

components

Out[156]:

svd0 svd1 svd2 svd3 svd4

Customer Lifetime Value 0.415830 0.469770 0.147810 0.007218 0.017383

Income -0.242604 0.782291 0.149417 0.116633 -0.052793

Monthly Premium Auto 0.615247 0.224945 0.024034 0.082746 -0.025281

Months Since Last Claim 0.019734 -0.183018 0.627694 0.081589 -0.269760

Months Since Policy Inception 0.019469 0.128531 -0.656189 0.016069 0.228362

Number of Open Complaints -0.036615 -0.104806 0.018225 0.961470 0.233382

Number of Policies 0.002970 -0.016142 0.355253 -0.219044 0.903810

Total Claim Amount 0.622561 -0.235534 -0.064253 0.013359 0.003624

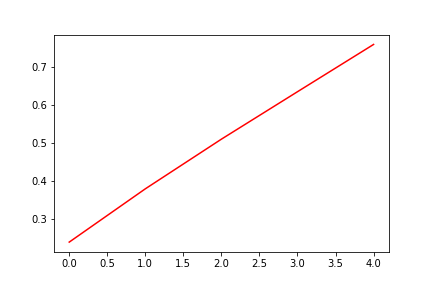
print(svd.explained\_variance\_ratio\_)

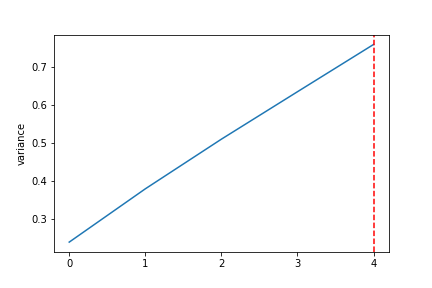
[0.23943177 0.14027837 0.13075081 0.12490888 0.12444049]

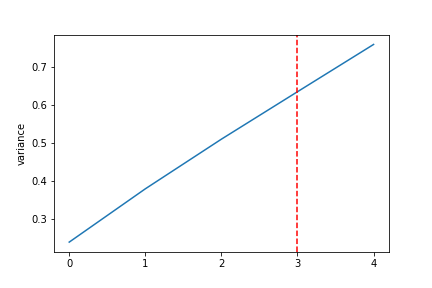
var1 = np.cumsum(svd.explained\_variance\_ratio\_)

print(var1)

[0.23943177 0.37971013 0.51046095 0.63536982 0.75981031]







final = pd.concat([df.Customer, svd\_res.iloc[:, 0:4]], axis = 1)

final.columns = ['Customer', 'svd0', 'svd1', 'svd2', 'svd3']

final

Out[167]:

Customer svd0 svd1 svd2 svd3

0 BU79786 -0.988514 -0.445305 1.738864 -0.108212

1 QZ44356 1.760618 -1.579777 0.391470 -0.999510

2 AI49188 0.767736 0.560896 0.406998 -0.210295

3 WW63253 0.750825 -0.946134 0.169339 -0.852223

4 HB64268 -1.355272 0.007026 -0.427308 -0.297180

... ... ... ... ...

9129 LA72316 -0.160202 2.181523 -0.395019 -0.182010

9130 PK87824 -0.543906 -0.812424 -0.078823 -0.349737

9131 TD14365 0.801777 -1.536623 -0.477437 2.647552

9132 UP19263 0.718288 -1.134879 2.092450 -0.324251

9133 Y167826 -0.435340 -0.921750 -2.338228 -0.491840

[9134 rows x 5 columns]

**Deployment of SVD using Flask**

