**Batch: HO-ML 1 Experiment Number: 06**

**Roll Number: 16010422234 Name: Chandana Galgali**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aim of the Experiment:** To demonstrate the use of PCA for dimensionality reduction.

**Program/ Steps:**

1. Refer to: <https://www.kaggle.com/code/avikumart/pca-principal-component-analysis-from-scratch> for the code and the description for the PCA implementation. Use VScode and Matplotlib to reproduce the results on your machine.

**Output/Result:**

**# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,**

**# THEN FEEL FREE TO DELETE THIS CELL.**

**# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON**

**# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR**

**# NOTEBOOK.**

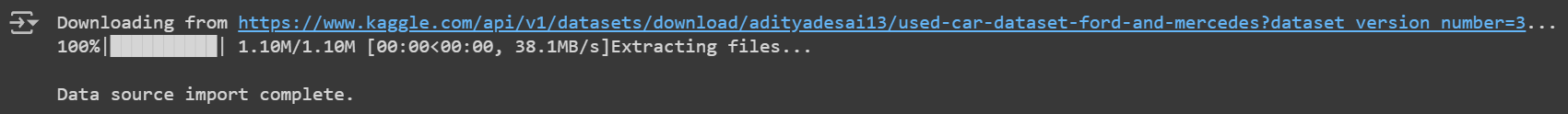
**import os**

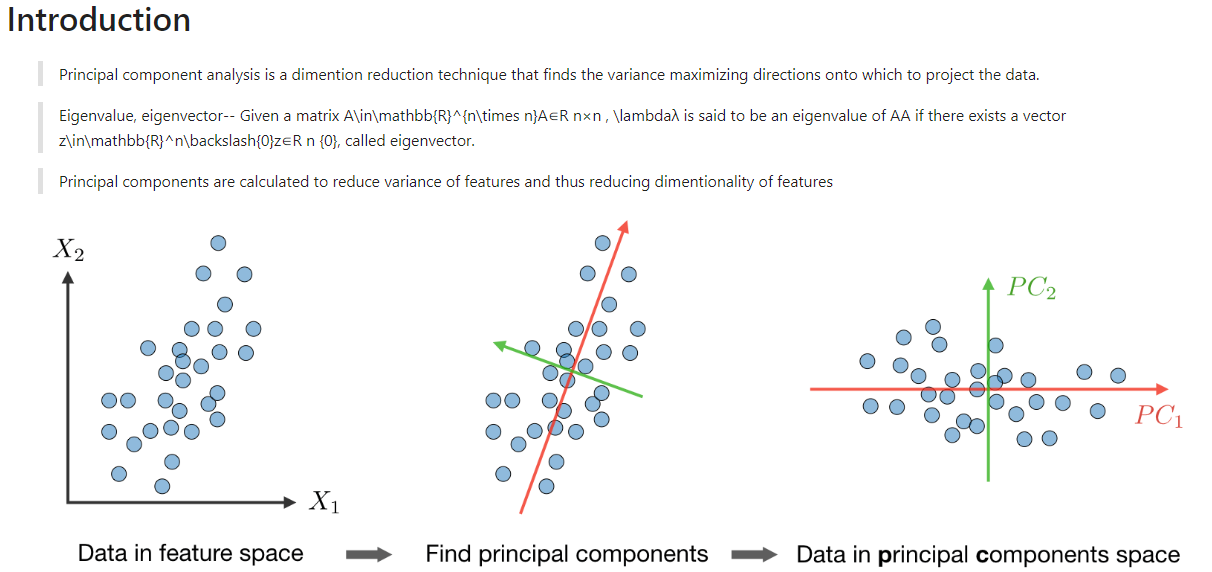
**import shutil**

**import kagglehub**

**adityadesai13\_used\_car\_dataset\_ford\_and\_mercedes\_path = kagglehub.dataset\_download('adityadesai13/used-car-dataset-ford-and-mercedes')**

**print('Data source import complete.')**

****

****

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.decomposition import PCA**

**from sklearn.feature\_selection import mutual\_info\_regression**

**# matplotlib defaults**

**plt.style.use("seaborn-darkgrid")**

**plt.rc("figure", autolayout=True)**

**plt.rc(**

**"axes",**

**labelweight="bold",**

**labelsize="large",**

**titleweight="bold",**

**titlesize=14,**

**titlepad=10,**

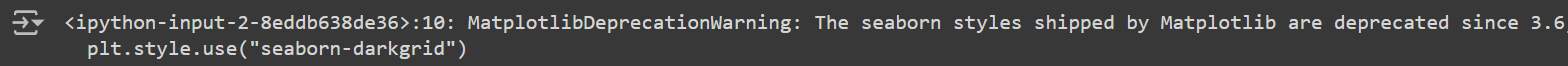
**)**

**import os**

**for dirname, \_, filenames in os.walk('/kaggle/input'):**

**for filename in filenames:**

**print(os.path.join(dirname, filename))**

****

**ford = pd.read\_csv('/content/ford.csv')**

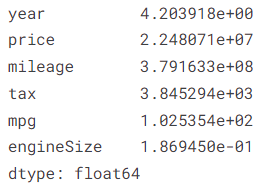
**ford**

|  | **model** | **year** | **price** | **transmission** | **mileage** | **fuelType** | **tax** | **mpg** | **engineSize** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **Fiesta** | **2017** | **12000** | **Automatic** | **15944** | **Petrol** | **150** | **57.7** | **1.0** |
| **1** | **Focus** | **2018** | **14000** | **Manual** | **9083** | **Petrol** | **150** | **57.7** | **1.0** |
| **2** | **Focus** | **2017** | **13000** | **Manual** | **12456** | **Petrol** | **150** | **57.7** | **1.0** |
| **3** | **Fiesta** | **2019** | **17500** | **Manual** | **10460** | **Petrol** | **145** | **40.3** | **1.5** |
| **4** | **Fiesta** | **2019** | **16500** | **Automatic** | **1482** | **Petrol** | **145** | **48.7** | **1.0** |
| **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** |
| **17960** | **Fiesta** | **2016** | **7999** | **Manual** | **31348** | **Petrol** | **125** | **54.3** | **1.2** |
| **17961** | **B-MAX** | **2017** | **8999** | **Manual** | **16700** | **Petrol** | **150** | **47.1** | **1.4** |
| **17962** | **B-MAX** | **2014** | **7499** | **Manual** | **40700** | **Petrol** | **30** | **57.7** | **1.0** |
| **17963** | **Focus** | **2015** | **9999** | **Manual** | **7010** | **Diesel** | **20** | **67.3** | **1.6** |
| **17964** | **KA** | **2018** | **8299** | **Manual** | **5007** | **Petrol** | **145** | **57.7** | **1.2** |

**17965 rows × 9 columns**

**# variance among numircal features**

**ford.var()**

****

**y = ford['price']**

**ford\_ = ford.drop('price', axis=1)**

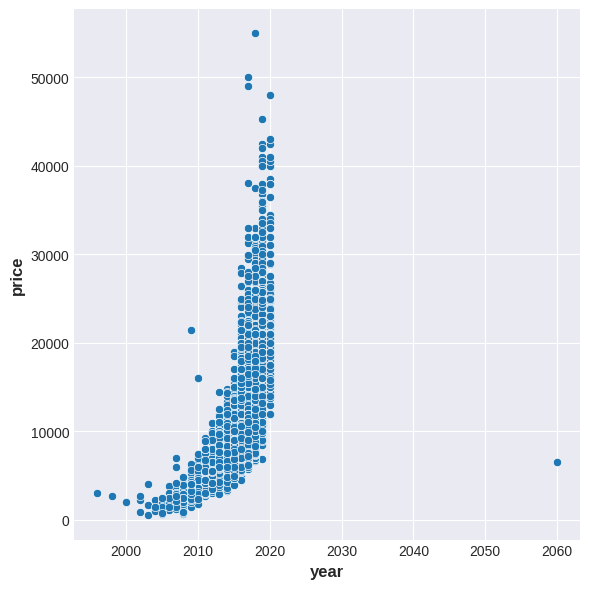
**cols = [col for col in ford\_.columns if ford\_[col].dtype in ['int64','float64']]**

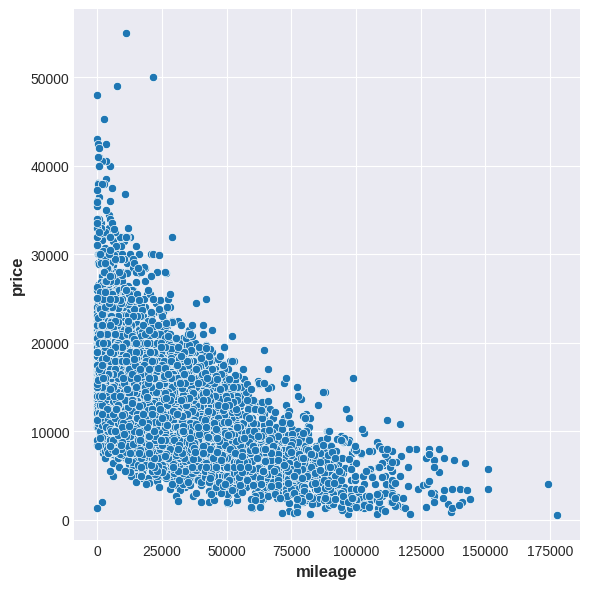
**for idx, col in enumerate(cols):**

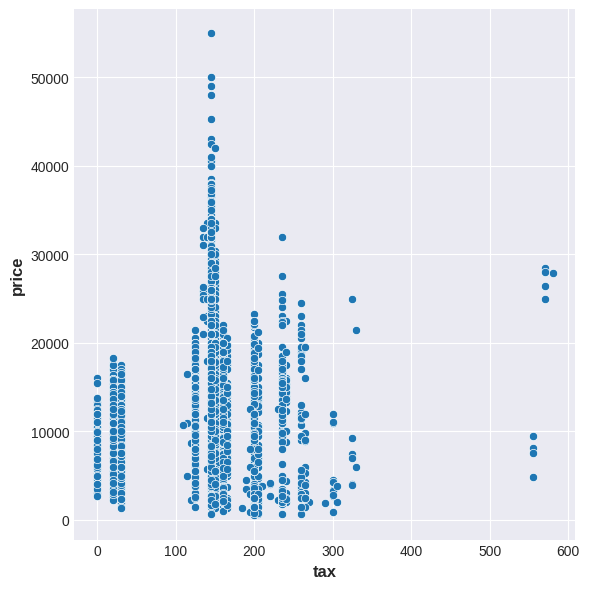
**plt.figure(idx, figsize=(6,6))**

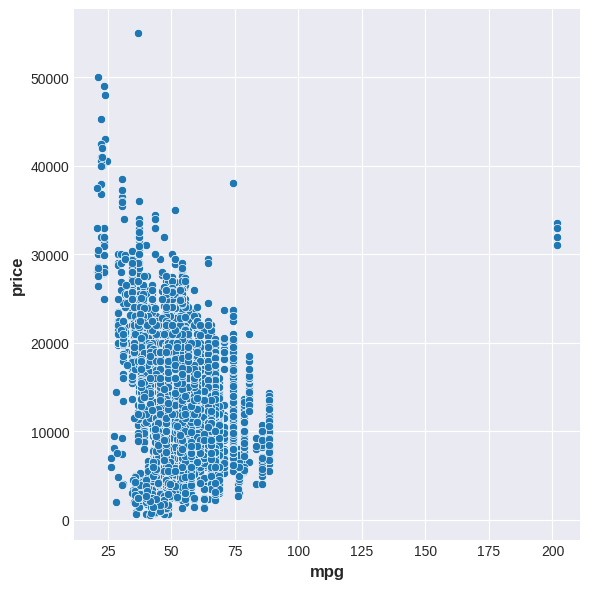
**sns.scatterplot(x=col, y=y, data=ford\_)**

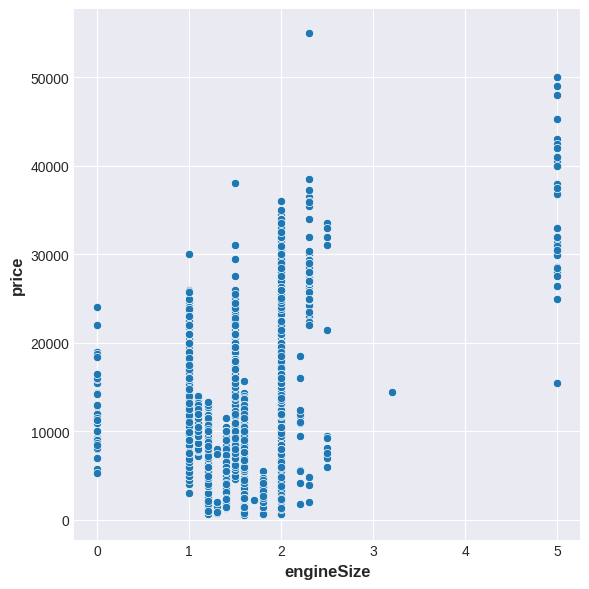
**plt.show**

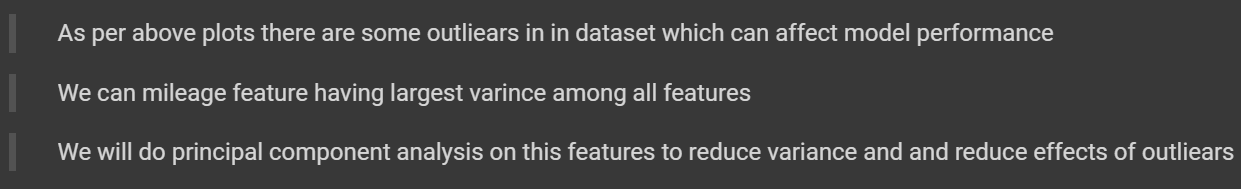
****

**\**

****

****

****

****

**features = ['mileage','year','mpg','tax','engineSize']**

**X = ford\_[features]**

**# normalizing features**

**X\_norm = (X - X.mean(axis=0))/X.std(axis=0)**

**# principal component analysis on features**

**pca = PCA()**

**# fit and transform X\_norm to PCA dataframe**

**X\_pca = pca.fit\_transform(X\_norm)**

**# converting to dataframe**

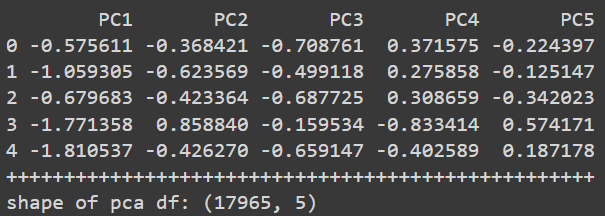
**names = [f"PC{i+1}" for i in range(X\_pca.shape[1])]**

**X\_pcadf = pd.DataFrame(X\_pca, columns=names)**

**print(X\_pcadf.head())**

**print("+++++++++++++++++++++++++++++++++++++++++++++++++++")**

**print("shape of pca df:", X\_pcadf.shape)**

****

**pca.singular\_values\_**

****

**X\_norm.T**

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **...** | **17955** | **17956** | **17957** | **17958** | **17959** | **17960** | **17961** | **17962** | **17963** | **17964** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **mileage** | **-0.381039** | **-0.733389** | **-0.560167** | **-0.662672** | **-1.123742** | **0.619777** | **-1.095650** | **-0.529456** | **-0.845806** | **1.272454** | **...** | **-0.837949** | **-0.950777** | **1.168819** | **-0.513793** | **0.630305** | **0.410041** | **-0.342214** | **0.890318** | **-0.839849** | **-0.942714** |
| **year** | **0.065075** | **0.552798** | **0.065075** | **1.040520** | **1.040520** | **-0.910370** | **1.040520** | **0.065075** | **1.040520** | **0.552798** | **...** | **0.552798** | **0.552798** | **-0.910370** | **1.040520** | **-0.422648** | **-0.422648** | **0.065075** | **-1.398093** | **-0.910370** | **0.552798** |
| **mpg** | **-0.020442** | **-0.020442** | **-0.020442** | **-1.738794** | **-0.909245** | **-0.988249** | **-0.741360** | **-0.356212** | **-1.551158** | **0.344955** | **...** | **-0.909245** | **-1.067254** | **-0.454968** | **-0.909245** | **1.085624** | **-0.356212** | **-1.067254** | **-0.020442** | **0.927615** | **-0.020442** |
| **tax** | **0.591279** | **0.591279** | **0.591279** | **0.510647** | **0.510647** | **0.510647** | **0.510647** | **0.510647** | **0.510647** | **0.510647** | **...** | **0.510647** | **0.510647** | **0.188121** | **0.510647** | **-1.505142** | **0.188121** | **0.591279** | **-1.343879** | **-1.505142** | **0.510647** |
| **engineSize** | **-0.811401** | **-0.811401** | **-0.811401** | **0.345012** | **-0.811401** | **0.576295** | **-0.811401** | **-0.348836** | **1.501425** | **-0.811401** | **...** | **-0.811401** | **-0.811401** | **-0.811401** | **-0.811401** | **0.345012** | **-0.348836** | **0.113730** | **-0.811401** | **0.576295** | **-0.348836** |

**5 rows × 17965 columns**

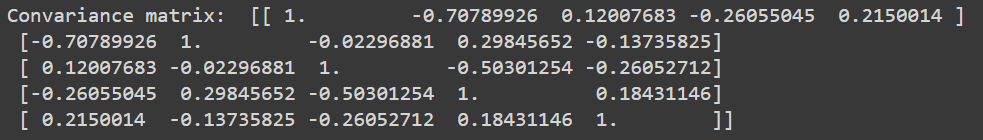
**# we will perform PCA from scratch using numpy library**

**# X\_norm is the Z-scoretransformed dataframe in our dataset**

**# convert cov\_matrix from the X\_norm**

**cov\_matrix = np.cov(X\_norm.T)**

**print("Convariance matrix: ", cov\_matrix)**

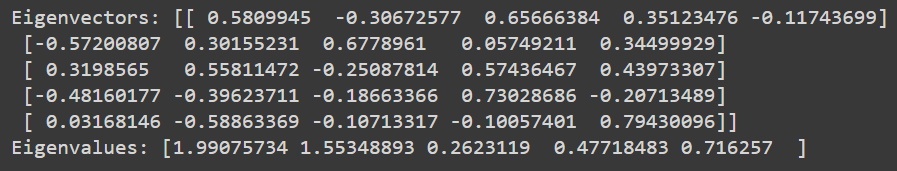
****

**# from COV\_MATRIX calculate eigenvectors and eigenvalues**

**eigenvalues, eigenvectors = np.linalg.eig(cov\_matrix)**

**print("Eigenvectors:", eigenvectors)**

**print("Eigenvalues:", eigenvalues)**

****

**# sort the eigen values and eigen vectors in descending order**

**eig\_pairs = [(eigenvalues[index],eigenvectors[:,index]) for index in range(len(eigenvalues))]**

**# sort the pairs**

**eig\_pairs.sort()**

**# reverse to make it in correct order**

**eig\_pairs.reverse()**

**print(eig\_pairs)**

**# extract the sorted eiganvalues and eiganvectors**

**eigenvalues\_sorted = [eig\_pairs[index][0] for index in range(len(eigenvalues))]**

**eigenvectors\_sorted = [eig\_pairs[index][1] for index in range(len(eigenvalues))]**

**# print sorted eigan values**

**print("Sorted eigan values:", eigenvalues\_sorted)**

**[(1.9907573430794159, array([ 0.5809945 , -0.57200807, 0.3198565 , -0.48160177, 0.03168146])), (1.5534889266247038, array([-0.30672577, 0.30155231, 0.55811472, -0.39623711, -0.58863369])), (0.7162570002651933, array([-0.11743699, 0.34499929, 0.43973307, -0.20713489, 0.79430096])), (0.47718482565372217, array([ 0.35123476, 0.05749211, 0.57436467, 0.73028686, -0.10057401])), (0.26231190437677343, array([ 0.65666384, 0.6778961 , -0.25087814, -0.18663366, -0.10713317]))]**

**Sorted eigan values: [1.9907573430794159, 1.5534889266247038, 0.7162570002651933, 0.47718482565372217, 0.26231190437677343]**

**print(eigenvectors\_sorted)**

**[array([ 0.5809945 , -0.57200807, 0.3198565 , -0.48160177, 0.03168146]), array([-0.30672577, 0.30155231, 0.55811472, -0.39623711, -0.58863369]), array([-0.11743699, 0.34499929, 0.43973307, -0.20713489, 0.79430096]), array([ 0.35123476, 0.05749211, 0.57436467, 0.73028686, -0.10057401]), array([ 0.65666384, 0.6778961 , -0.25087814, -0.18663366, -0.10713317])]**

**# plot the variance plots using sorted eigenvalues and eigenvectors**

**total = sum(eigenvalues\_sorted)**

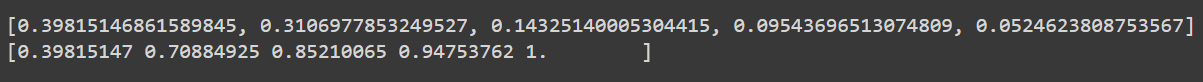
**var\_explained = [(i/total) for i in eigenvalues\_sorted]**

**# calculate cumulative variance**

**cum\_var\_exp = np.cumsum(var\_explained)**

**print(var\_explained)**

**print(cum\_var\_exp)**

****

**# transforming original dataframe into PCA**

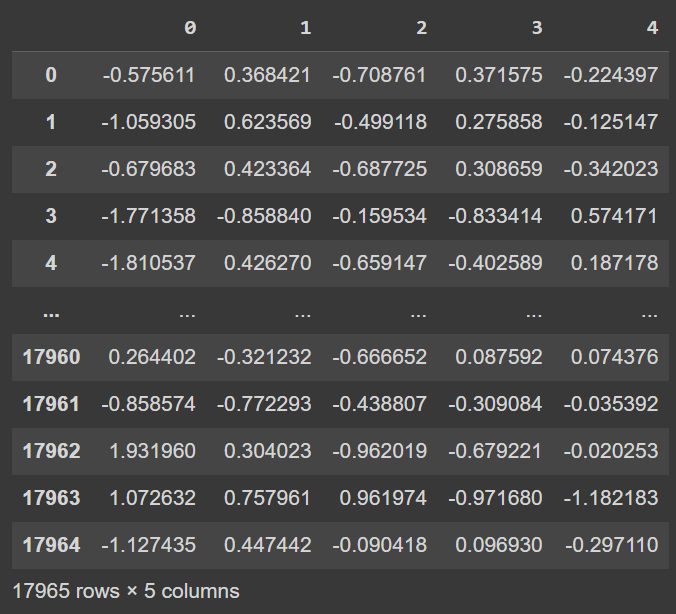
**vect = np.array(eigenvectors\_sorted)**

**# dot product to create principal components analysis**

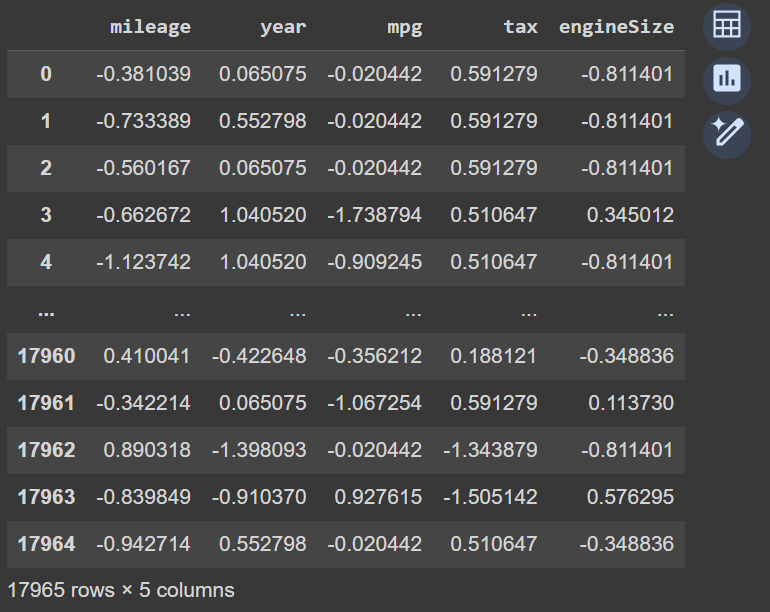
**X\_vect\_pca = np.dot(X\_norm,vect.T)**

**# pca dataframe**

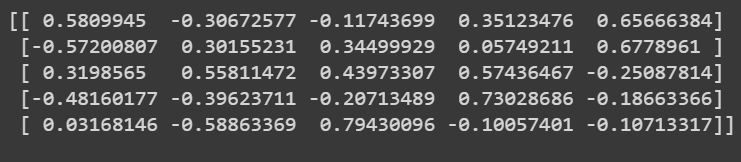
**pd.DataFrame(X\_vect\_pca)**

****

**X\_norm**

****

**print(vect.T)**

****

**evr = pca.explained\_variance\_ratio\_**

**print(evr)**

**features = ['mileage','year','mpg','tax','engineSize']**

**# plot the EVR using matplotlib pyplot**

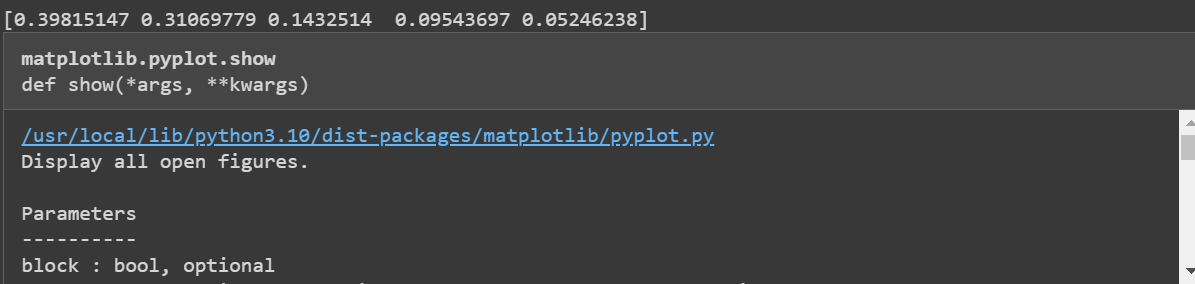
**plt.figure(figsize=(6,6))**

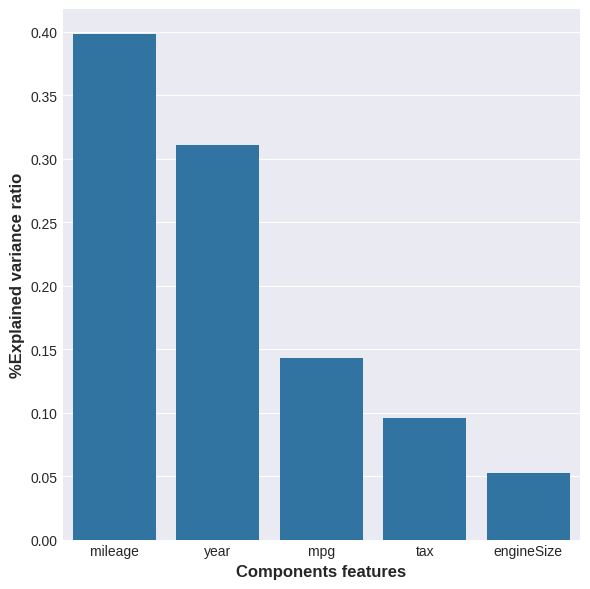
**sns.barplot(x=np.array(features), y=evr)**

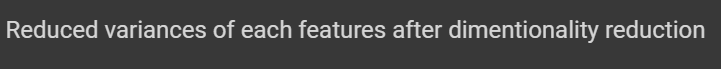
**plt.xlabel("Components features")**

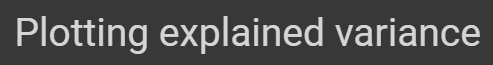
**plt.ylabel("%Explained variance ratio")**

**plt.show**

****

****

****

****

**ev = pca.explained\_variance\_**

**print(ev)**

**features = ['mileage','year','mpg','tax','engineSize']**

**plt.figure(figsize=(6,6))**

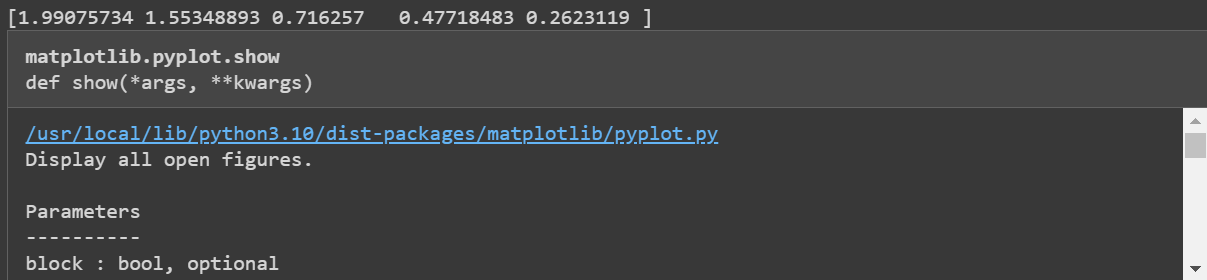
**sns.lineplot(x=np.array(features), y=ev)**

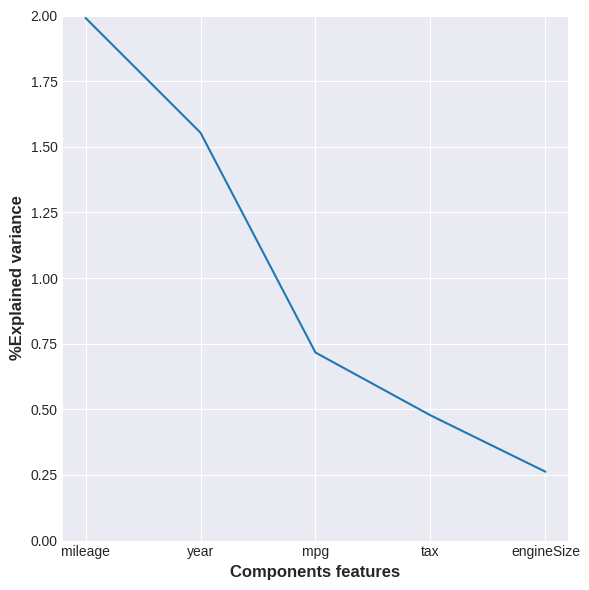
**plt.xlabel("Components features")**

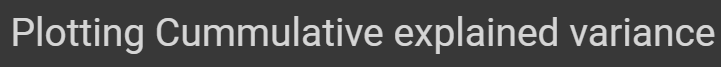
**plt.ylabel("%Explained variance")**

**plt.ylim(0,2)**

**plt.show**

****

****

****

**evc = np.cumsum(pca.explained\_variance\_)**

**print(evc)**

**features = ['mileage','year','mpg','tax','engineSize']**

**plt.figure(figsize=(6,6))**

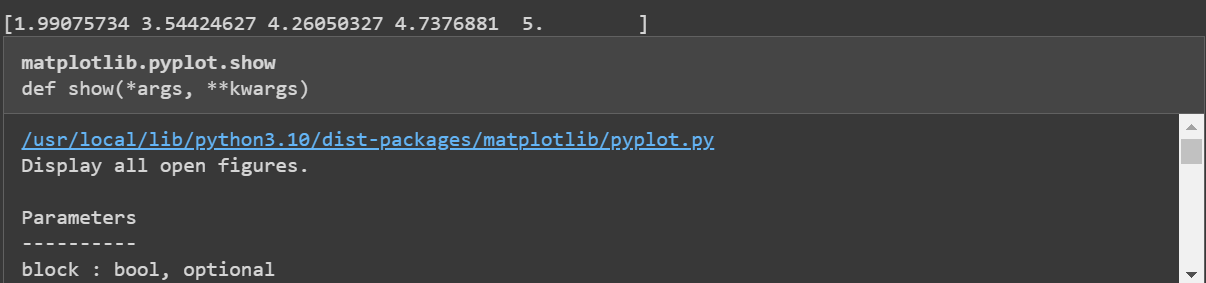
**sns.lineplot(x=np.array(features), y=evc)**

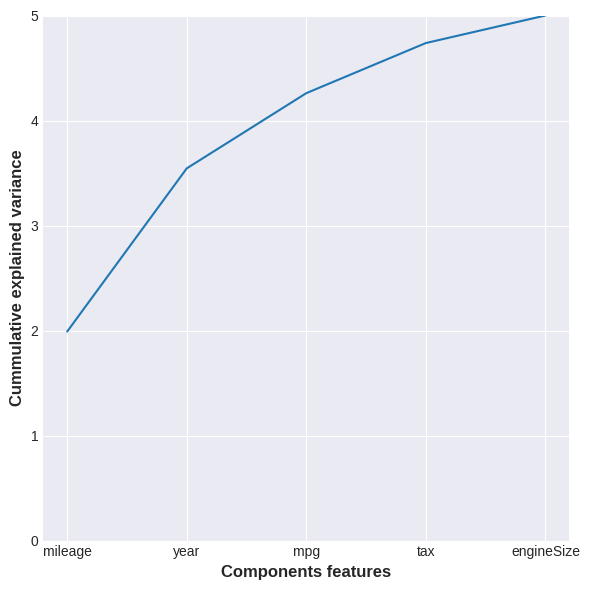
**plt.xlabel("Components features")**

**plt.ylabel("Cummulative explained variance")**

**plt.ylim(0,5)**

**plt.show**

****

****

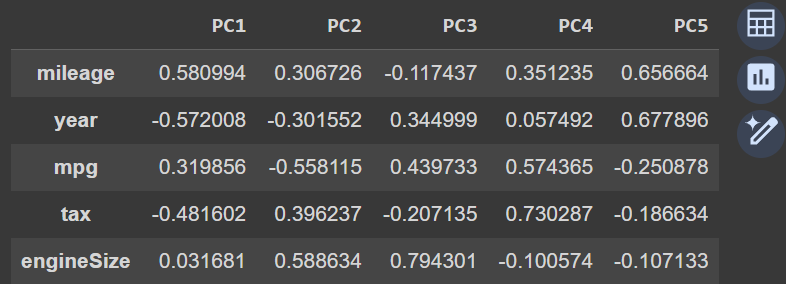
****

**loadings = pd.DataFrame(pca.components\_.T ,**

**index=np.array(features),**

**columns=names)**

**loadings**

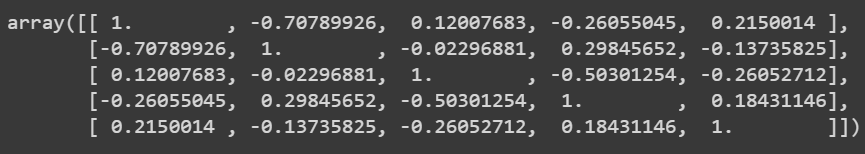
****

**pca.noise\_variance\_**

****

**# covariance matrix of principal components**

**pca.get\_covariance()**

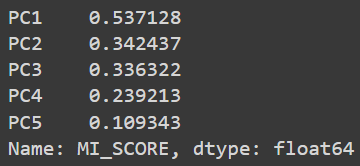
****

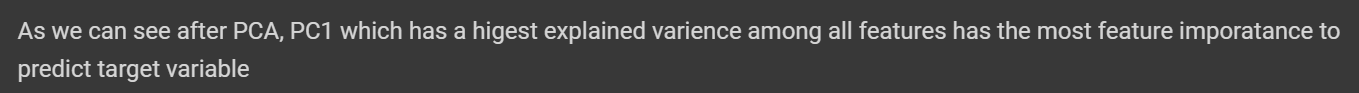
**y = ford['price']**

**mi\_score = mutual\_info\_regression(X\_pcadf,y, discrete\_features=False)**

**mi\_score = pd.Series(mi\_score, index=X\_pcadf.columns, name="MI\_SCORE")**

**print(mi\_score)**

****

****

**Post Lab Question-Answers:**

**1. Explain the term Eigenvalues in your own words.**

Eigenvalues are numbers that give us information about how much variance exists in the data along the direction of its corresponding eigenvector. When performing Principal Component Analysis (PCA), eigenvalues represent the magnitude of the variance captured by each principal component. The larger the eigenvalue, the more important that component is in explaining the variability of the data.

**Outcomes: Comprehend radial-basis-function (RBF) networks and Kernel learning method**

**Conclusion (based on the Results and outcomes achieved):**

In this experiment, the Principal Component Analysis (PCA) method was successfully applied to reduce the dimensionality of a dataset. By identifying and focusing on the principal components with the highest eigenvalues, we managed to retain the most significant information while removing less important features, which reduced the dataset's complexity. The reduction in dimensions made the data more manageable without losing crucial patterns, thus enhancing computational efficiency. The results achieved the objective of dimensionality reduction with minimal information loss.

**References:**

Books/ Journals/ Websites:

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3nd Edition