FODS ASSIGNMENT - 2

Submitted To

Dr. NL Bhanu Murthy

Foundations of Data Science: CS F320

Submitted By:

Name	ID NO
AASHISH CHANDRA K	2021A7PS0467H
AASHUTOSH A V	2021A7PS0056H
TUSHAR BRIJESH CHENAN	2020A7PS0253H

Birla Institute of Technology and Science, Hyderabad Campus

Aim of the project

This project aims to deepen the understanding of **Principal Component Analysis** (PCA) through the implementation of PCA using NumPy and Pandas libraries on two distinct datasets, namely *audi.xlsx* and *hitters.csv*. For Task A, the focus is on dimensionality reduction and visualization of principal components in the *audi* dataset, employing step-by-step PCA implementation. Task B involves conducting PCA on the *hitters* dataset, determining the optimal number of components for efficient prediction using Mean Squared Error (MSE) or Root Mean Squared Error (RMSE), and testing the most efficient model. Through comprehensive documentation and presentation, the project aims to provide insights into the significance of PCA in capturing variance, reducing dimensionality, and enhancing prediction efficiency.

Acknowledgment

We would like to express our sincere gratitude to Prof. N L Bhanu Murthy for imparting invaluable knowledge and guidance throughout our foundation of data science course. His excellent class lectures and commitment to clarifying our concepts have been instrumental in shaping our understanding of complex topics, including Principal Component Analysis (PCA).

We extend our appreciation to the dedicated PhD Teaching Assistants for their time and effort in evaluating our assignments. Their constructive feedback and insightful comments have significantly contributed to our learning experience.

Furthermore, we would like to thank everyone who has provided constant support and encouragement throughout this project. Your assistance has been pivotal in overcoming challenges and achieving a deeper understanding of the intricacies involved in implementing PCA.

Task A: Implementing PCA from Scratch and Applying it to Audi.xlsx dataset

Task:

1. Data Understanding and Representation:
    a. Import *audi.xlsx* and represent features as a matrix (rows: cars, columns: features).
2. Implementing PCA using Covariance Matrices:
    a. Calculate feature means, center the dataset, and compute the covariance matrix.
3. Eigenvalue-Eigenvector Equation:
    a. Formulate and solve the eigenvalue-eigenvector equation using NumPy.
4. Solving for Principal Components:
    a. Implement a method using NumPy to find solutions and select top *k* eigenvectors.
5. Sequential Variance Increase:
    a. Calculate total variance covered by principal components and analyze sequential cumulative increase.
6. Visualization using Pair Plots:
    a. Plot pair plots of original features and project principal components onto these plots.
7. Conclusion and Interpretation:
    a. Interpret PCA results, discuss the significance of principal components, and analyze dimensionality reduction effectiveness.
8. Documentation and Presentation:
    a. Provide a well-documented report with steps, derivations, and code.

Methodology:

For importing the dataset, we can use the well documented *pandas.read_excel* method:

```
df = pd.read_excel("audi.xlsx")
```

We then drop unnecessary columns and remap non numeric columns into numeric data types.

We then standardized the dataset by transforming each feature into a mature feature with mean as $\mu = 0$ and $\sigma = 1$.

We then compute the covariance matrix with the *pandas.DataFrame.cov* method:

```
covariance_matrix = df.cov()
```

Then, we compute the eigenvalues and eigenvectors of the covariance matrix using the *np.linalg.eigh* function:

```
eigen_values, eigen_vectors = np.linalg.eigh(covariance_matrix)
```

Then, we can extract the top k principal components by looking at the eigenvalues and understanding that the fraction of variance captured by a particular eigenvalue λ_i is given by the value: $\frac{\lambda_i}{\sum_i \lambda_i}$

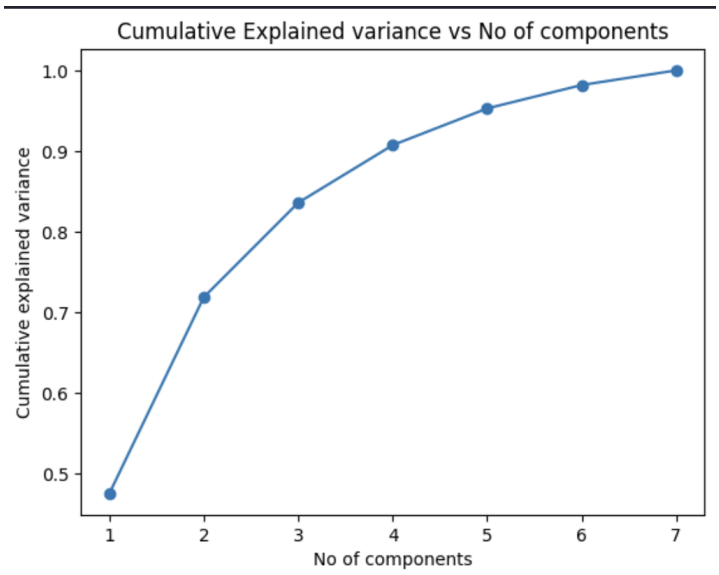
This formula is also used to track the running sum of the sequential variance increase as we increase the number of principal components.

Lastly, we plot the pair plots of the original features as well as principal components using the *seaborn.pairplo*t function.

Results and Observations:

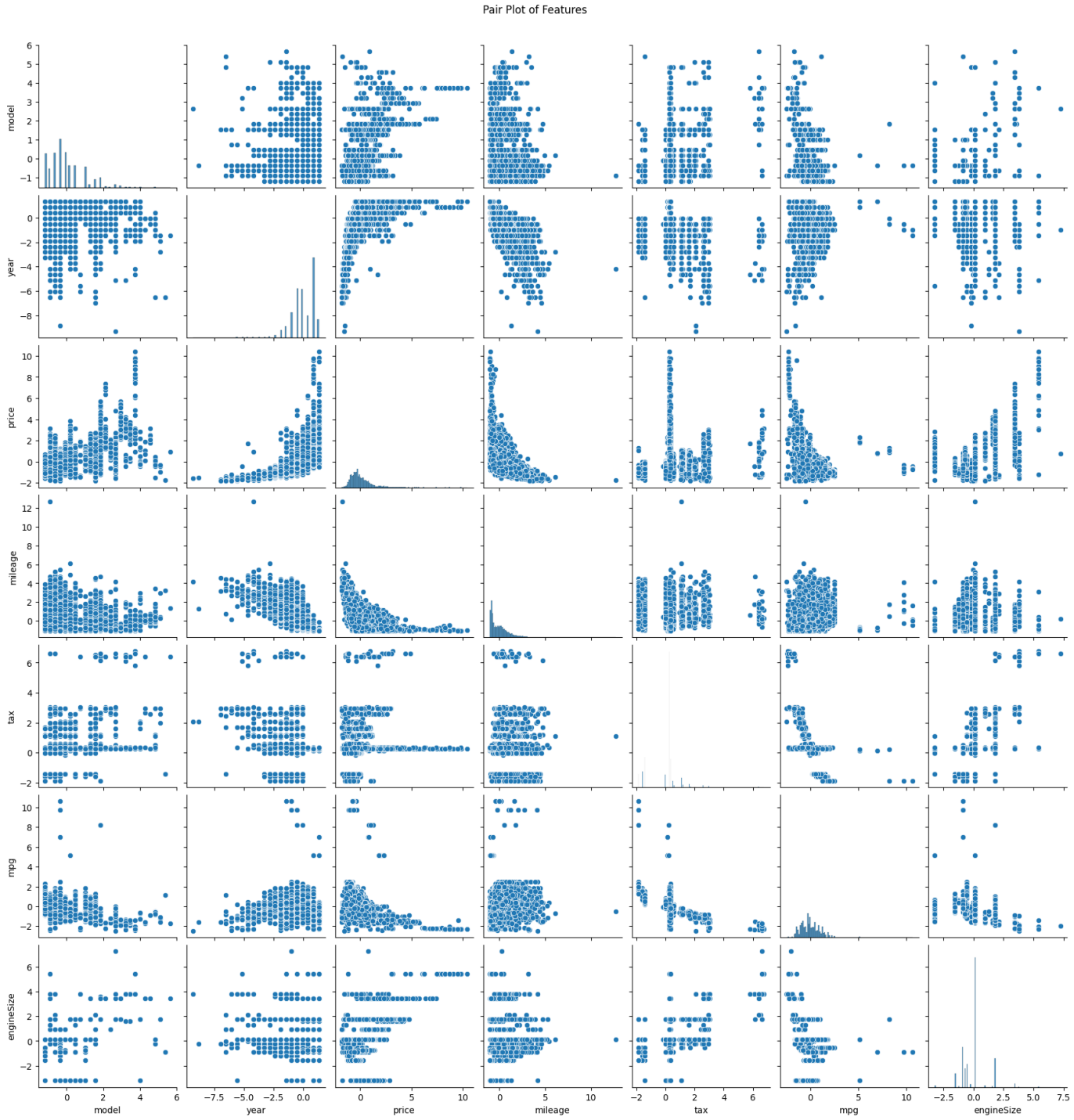
The heatmap of the covariance matrix indicates the correlation between various features of the dataset:

We then computed the eigenvalues and eigenvectors of the covariance matrix and them made the following plot:

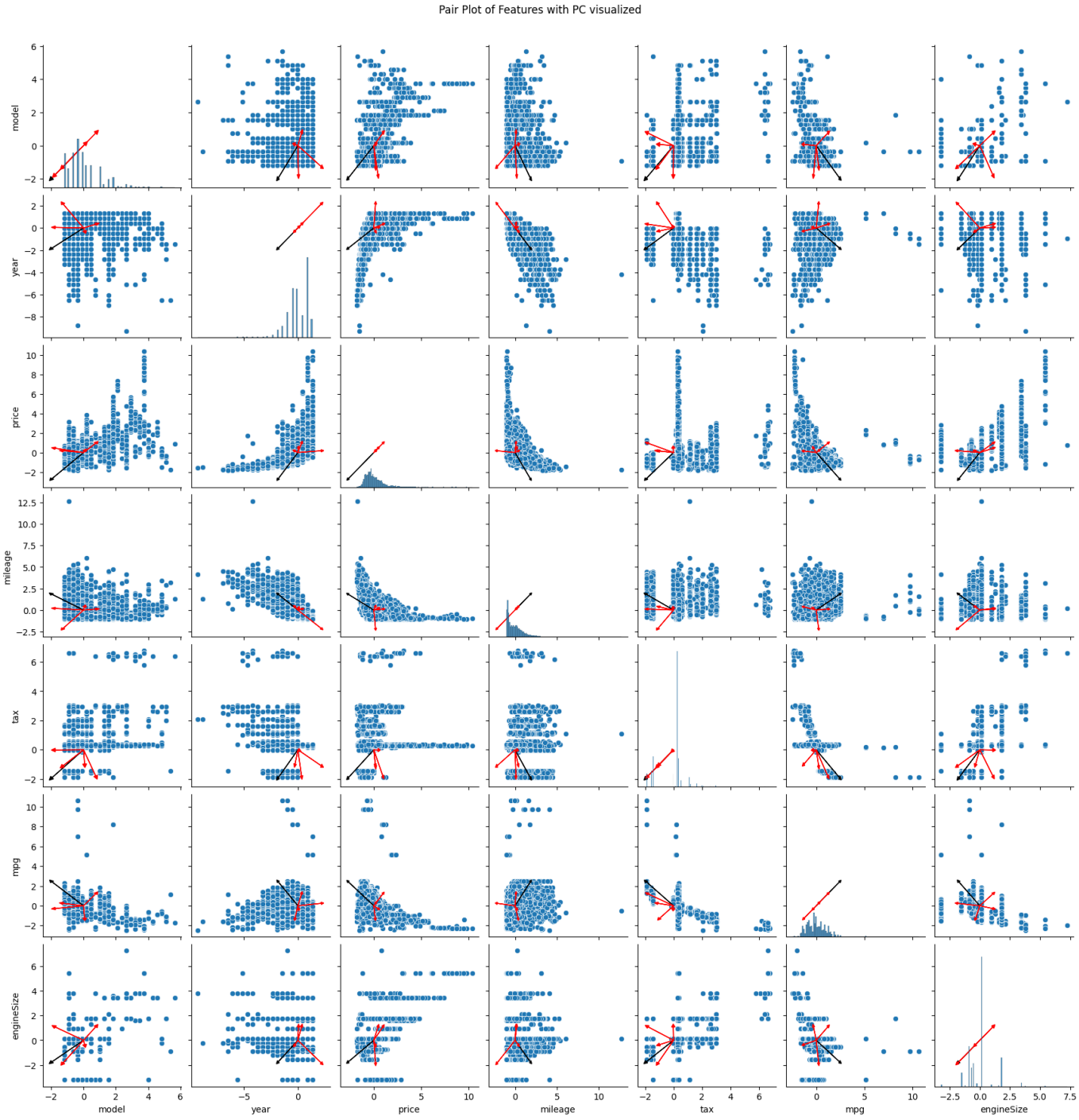


We can see that the contribution of the **first couple** of eigenvalues is very high to the total variance in comparison to the **later** eigenvalues. This is due to the fact that $\lambda_1 > \lambda_2 > \lambda_3 \cdots$

We then made a pair plot of the original feature prior to PCA:



Following this, we made a pairplot visualizing the PCA components on each of the pairplots:



Conclusions:

The conclusions from the first task are as follows:

1. The number of principal components is equal to the number of features taken into consideration from the dataset.
2. Projecting the original data points onto the principal components allows us to retain the maximum amount of variance in the original data as possible

a. Projecting onto the first principal component retains the highest amount of variance.
b. This is followed by the second, third component and so on.

Task B: PCA Analysis and Determining Optimal Number of Components

Task:

1. Exploratory Data Analysis (EDA)
    a. Load the *hitters* dataset and perform EDA to understand its structure, features, and relationships.
    b. Make sure to handle NULL values and eliminate any unwanted columns or data inconsistencies.
2. PCA Analysis
    a. Apply PCA on the cleaned dataset to reduce dimensionality.
    b. Determine the number of principal components required for efficient prediction.
3. Model Training and MSE/RMSE Calculation
    a. Split the dataset into training and testing sets.
    b. For each number of principal components considered, build a regression model using those components.
    c. Calculate the MSE or RMSE for each model on the test set to assess prediction efficiency
4. Plotting Number of Components vs RMSE
    a. Plot a graph of the number of components against RMSE to visualize the relationship.
    b. Identify the point where RMSE reaches a minimum or starts stabilizing, indicating an efficient number of components.
5. Testing the Most Efficient Model
    a. Select the model with the optimal number of components based on the graph.
    b. Test the selected model by predicting a specific point and providing its predicted y value.
6. Conclusion and Analysis
    a. Interpret the graph of the number of components vs RMSE to identify the most efficient model.
    b. Discuss the significance of selecting an appropriate number of components for prediction efficiency
    c. Analyze the predicted value (y_pred) from the chosen model and its significance.
7. Documentation and Presentation
    a. Present a comprehensive report detailing the steps taken, model implementations, results, and analysis.
    b. Include the graph illustrating the number of components vs RMSE and the interpretation of the findings.
    c. Prepare a clear and concise presentation summarizing the key findings for a broader audience.

Methodology:

First, we import the dataset using the *pandas.read_excel* function:

```
df = pd.read_excel('Hitters.xlsx')
```

Then, we check if any columns have NULL values using the pandas.DataFrame.isna function:

```
df.isna().sum(), len(df)
```

We have identified that the *Salary* feature has null values. Therefore, we fill the salary feature with the mean of the columns with non-NULL values:

```
sal_mean = df.Salary.mean()
df["Salary"] = df.Salary.fillna(sal_mean)
```

We then drop the unnecessary columns and standardize the dataset:

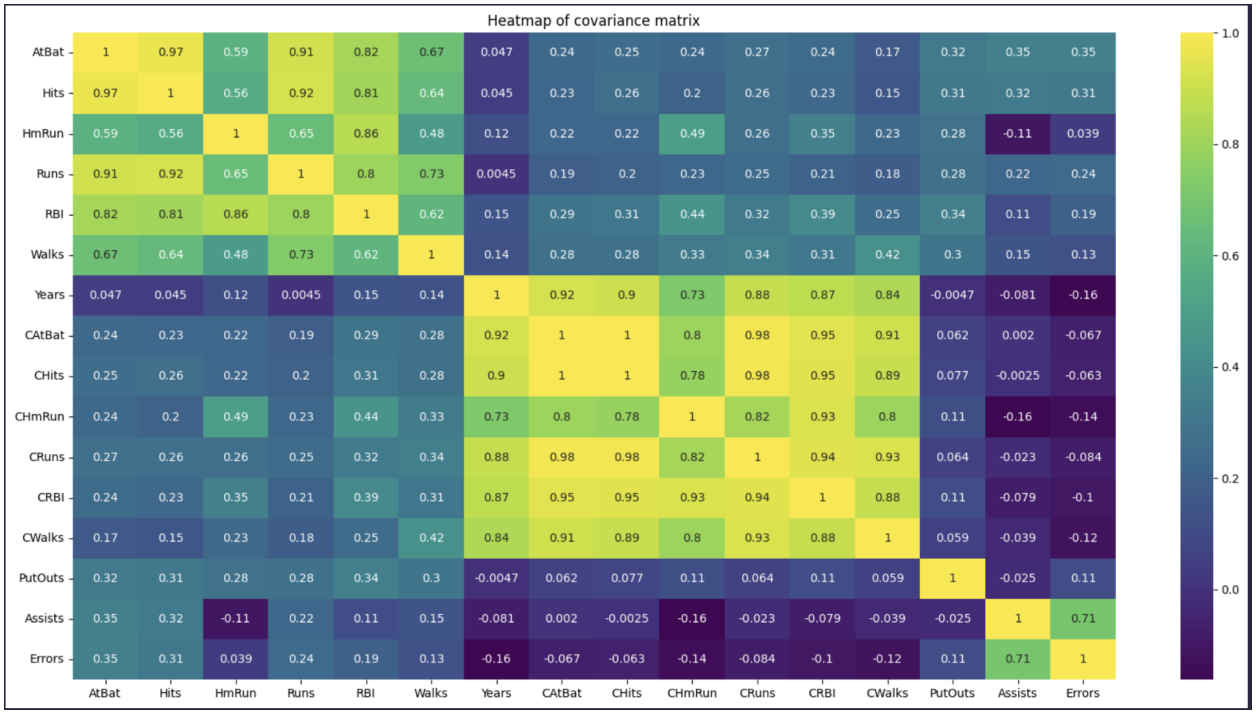
```
df.drop(columns=["League", "Division", "NewLeague"], axis=1, inplace=True)
shuffled_df = df.sample(frac=1, random_state=69)
X = shuffled_df.drop(["Salary"], axis=1)
y = shuffled_df["Salary"]

X_standardized = X.copy()
for col in list(X):
    X_standardized[col] = (X_standardized[col] -
X_standardized[col].mean()) /  X_standardized[col].std()
```

Next, we compute the covariance matrix using the *pandas.DataFrame.cov* function. Following this, we compute the eigenvalues and eigenvectors of the covariance matrix using the *np.linalg.eigh* function. We compute the total variance as the **sum** of eigenvalues and the fraction of variance captured by each eigenvector is defined as $\frac{\lambda_i}{\sum_i \lambda_i}$.

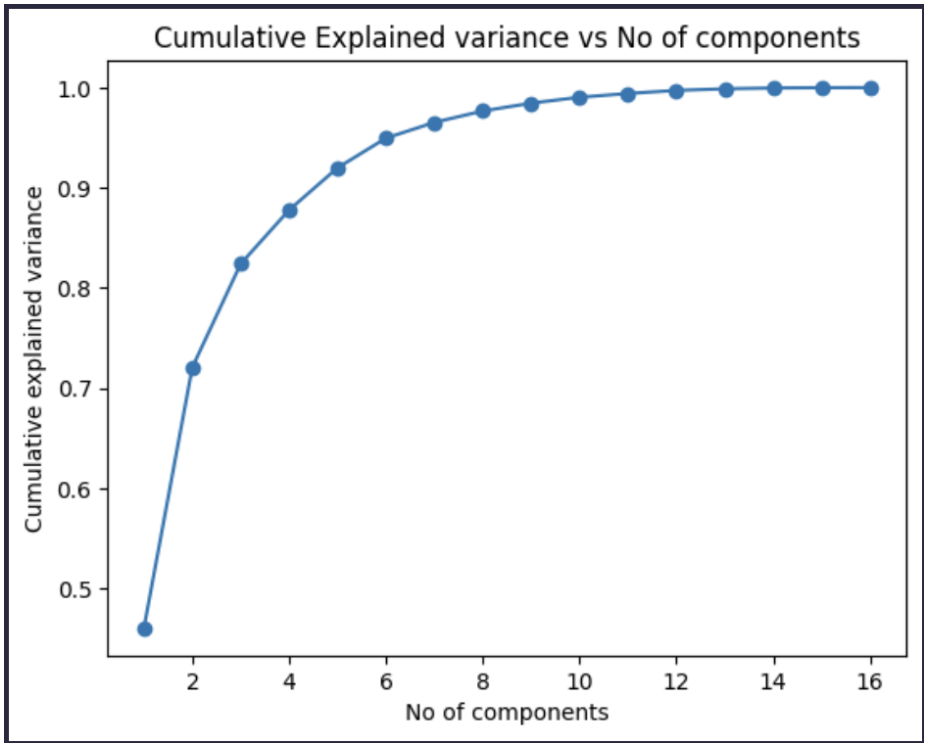
We then split the dataset into test and train data and then train the models using 1 to 16 components of the PCA(s) we have computed. For each set of components, we plot the RMSE and compare.

Results and Observations:

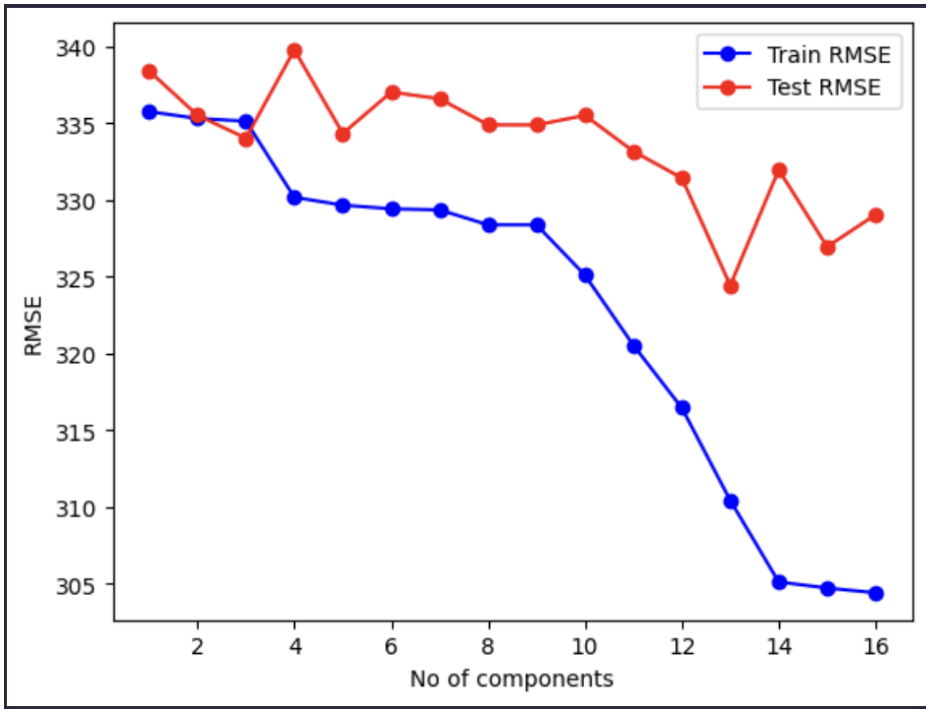


Here, we can see which features are highly correlated among themselves and we can use this to throw out any unnecessary / redundant features.

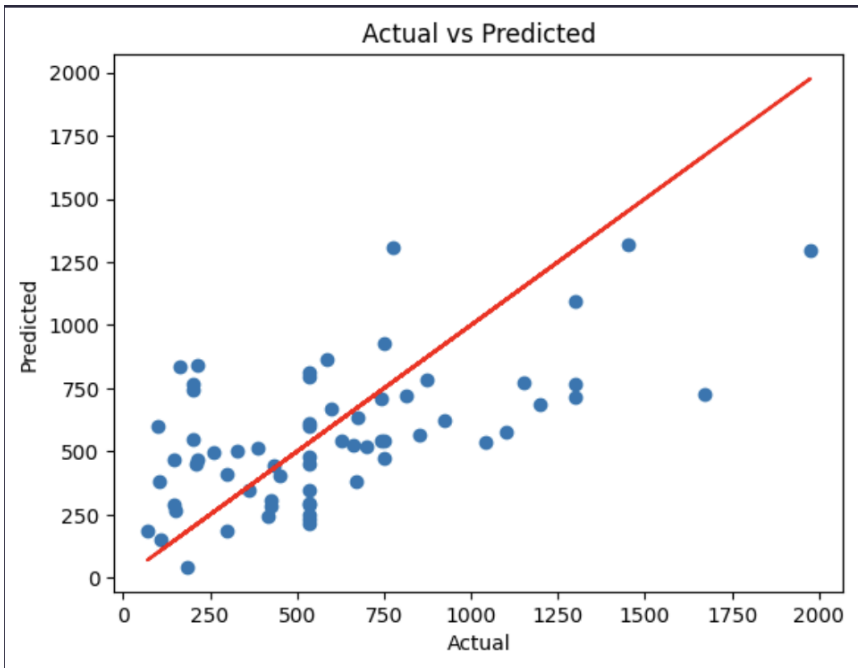
After this stage, 16 features are left. We have performed PCA on these 16 features and the graph below shows the variance captured as we increased the number of components.



We have considered all these features one by one and trained models following a $8:2$ split. We have plotted the RMSE of the models following the number of components:



We have used this plot to identify the optimal number of components. Using this information, we have trained the model using the optimal number of components and used it to generate a final predictive model for the data, which we have visualized using the following plot:



Conclusion:

The conclusion from the second task are as follows:

1. All 16 features seem to be relevant with respect to predicting the target feature.
2. After performing the PCA with 16 components, we are able to achieve 90% variance around 5 components.
3. The optimal set of components that gives us the lowest train MSE is the first 13 components of the PCA.
4. Using these 13 components, we can fit the best fit line on the data with the least RMSE error.
5. The red line in the graph represents where the predictions should've been, and that's the reference for the calculation of RMSE.
6. Evidently it also does make sense that the RMSE decreases as our number of components increases, since we're allowing the model to learn on extra distinguishable features. But the training graph itself doesn't move drastically after 13 components, which suggests that it is an optimal number of components for this study.