Lab Assignment 7
Prakhar Gupta
B21AI027

Question 1:

Part 1-

- Downloaded dataset using **wget** command called using **os.system**
- Using **wget** only downloaded the **anneal.names** then using **mv** command via **os.system** change the file name to **anneal_names.txt**
- Printed the content of **anneal_names.txt** then using the information present formed a **feature** array

```
features= ['family', 'product-type', 'steel', 'carbon',
'hardness', 'temper_rolling', 'condition', 'formability',
'strength', 'non-ageing', 'surface-finish', 'surface-quality',
'enamelability', 'bc', 'bf', 'bt', 'bw/me', 'bl', 'm', 'chrom',
'phos', 'cbond', 'marvi', 'exptl', 'ferro', 'corr',
'blue/bright/varn/clean', 'lustre', 'jurofm', 's', 'p', 'shape',
'thick', 'width', 'len', 'oil', 'bore', 'packing','classes']
```

- Added these **features** names to the df
- Loaded the **.csv** file into df using **pd.read_csv**
- Loaded two df one **anneal** and the other **anneal_test**

Part 2-

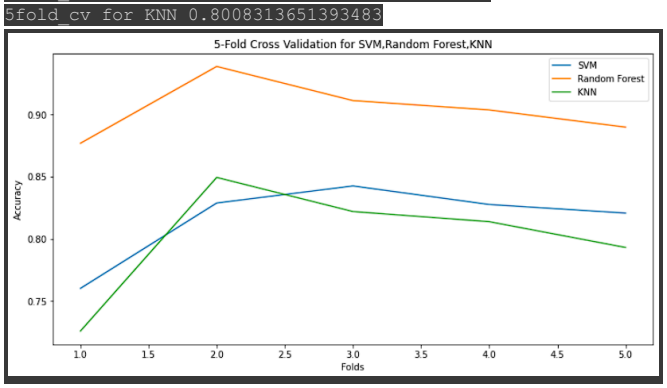
- Using **df.replace** replace all the '**?**' with **<NA>**
- Checked for not filled rows using **df.isnull().sum()**
- Dropped all the columns which has **<NA>** greater than 0.8*number of samples
- Then using **df.dropna** dropped rows which were having any null values
- Used **df.describe()** to get insights about the dataset
- Used **sns.countplot** to visualise the dataset for every feature
- Plotted **heatmap** of **correlation matrix**
- Did **label encoding** on **['product-type','steel','shape','classes']**
- Using **df.describe** we see that column **'product-type'** has same value for all examples , so we just dropped this column
- Coverted **df to X,y**
- Performed **StandardScaler()**
- Splitted the dataset into **65:35**

#unfe stands for not standard featurised dataset

Part 3-

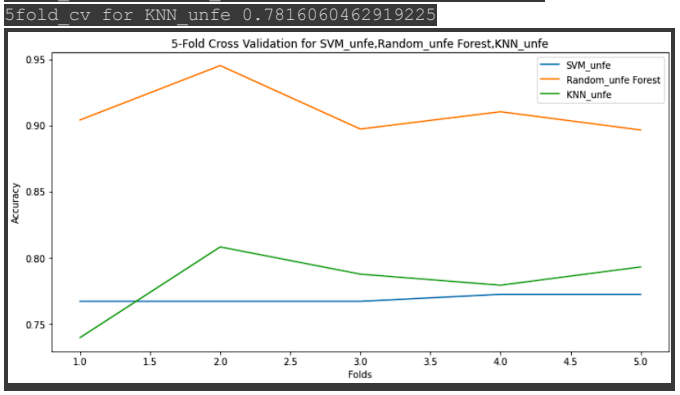
```
#For feature standardization X
```

- Used 5 fold cross validation on SVC
- ```
  5fold_cv for SVM 0.8159565422768068
  5fold_cv for Random Forest 0.9038261691072271
  5fold_cv for KNN 0.8008313651393483
  ```




```
#For not feature standardization X_unfe
```

- Used 5 fold cross validation on SVC
- ```
  5fold_cv for SVM_unfe 0.769239489844119
  5fold_cv for Random_unfe Forest 0.9106943788379784
  5fold_cv for KNN_unfe 0.7816060462919225
  ```



- SVMs are good in handling large-features datasets, and we had 9 features in our dataset, SVMs can also apply kernel tricks.
- Random Forest can handle large datasets with multiple features. It also avoids overfitting by training multiple decision trees.
- KNN: It can handle multi-class classification problems. KNN can also handle noisy data

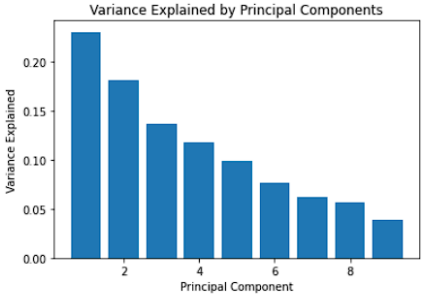
Part 4-

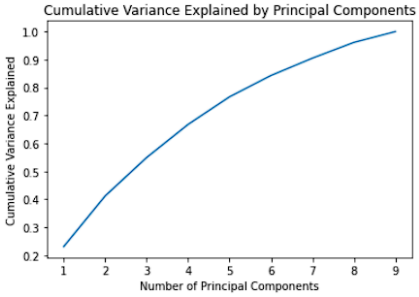
- Implemented Scratch built PCA with functions
- **fit,principal_component_scratch,covariance_matrix_scratch,eigen_valuesratch**
- It also Centralize the Data via feature-wise means and standard deviations.

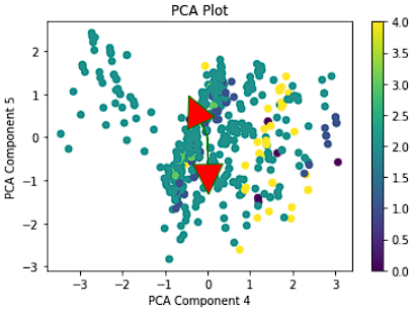
- Eigenvectors are used in PCA for transforming bases, when we try to find new bases in which the data features has zero covariance then we get that it is possible when our new base is the matrix of eigenvectors

Part 5-

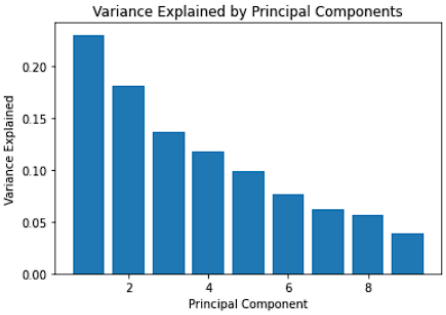
```
#For feature standardization X
```

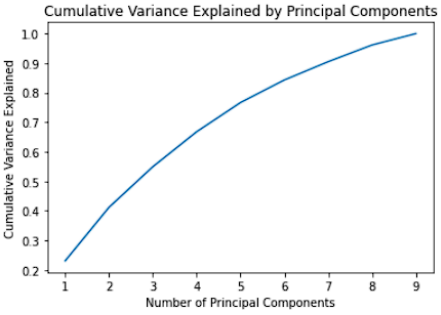


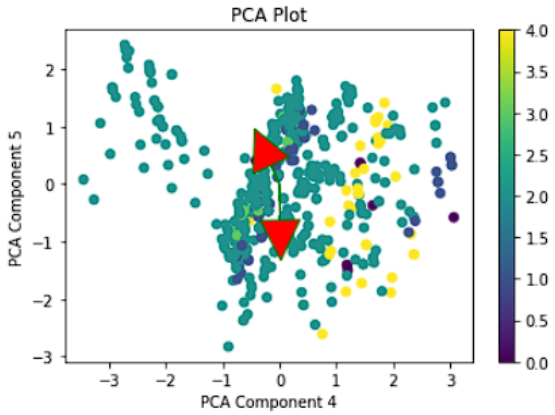




```
#For not feature standardization X_unfe
```



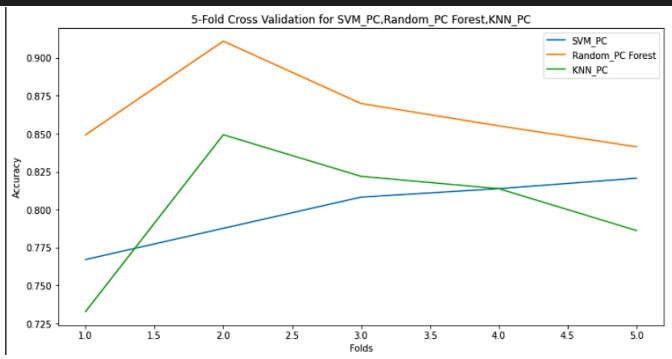




- The plots of X and X_unfe is same because inside PCA_scratch we normalised the data by first centering then divide by std

Part 6-

```
#For Principal Component
```



- 5fold_cv for SVM_PC 0.7994992914501653
  5fold_cv for Random_PC Forest 0.8653377420878602
  5fold_cv for KNN_PC 0.8008313651393483

```
Accuracy:
SVM: 0.8
Random Forest: 0.8117647058823529
KNN: 0.7725490196078432

Precision:
SVM: 0.7320983475860616
Random Forest: 0.8081541407031602
KNN: 0.7277256101217245

Recall:
SVM: 0.8
Random Forest: 0.8117647058823529
KNN: 0.7725490196078432

F1-score:
SVM: 0.7422053950609263
Random Forest: 0.8089453923924328
KNN: 0.7448697858928038
```

```
#For Feature Standarised X
```

```
Accuracy:
SVM: 0.8235294117647058
Random Forest: 0.8862745098039215
KNN: 0.7843137254901961

Precision:
SVM: 0.7639354748991795
Random Forest: 0.8920809290531577
KNN: 0.7622463274034587

Recall:
SVM: 0.8235294117647058
Random Forest: 0.8862745098039215
KNN: 0.7843137254901961

F1-score:
SVM: 0.7762194622628634
Random Forest: 0.8877465024144812
KNN: 0.7590840173641967
```

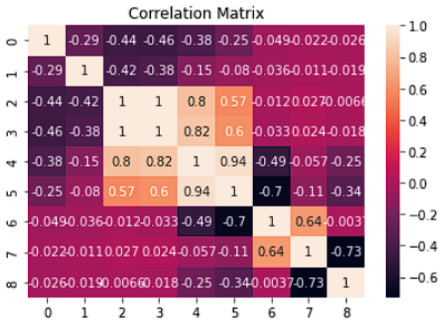
```
#For not Feature Standarised X_unfe
```

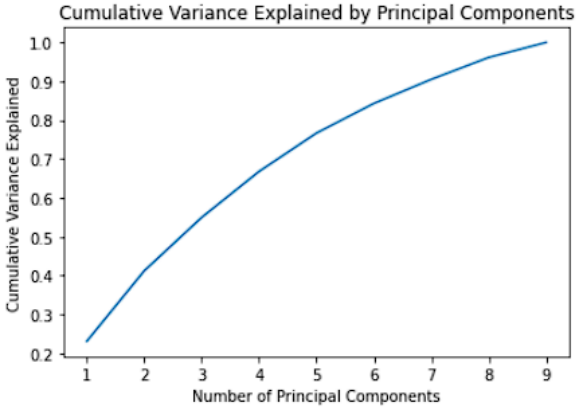
```
Accuracy:
SVM: 0.7686274509803922
Random Forest: 0.8901960784313725
KNN: 0.7411764705882353

Precision:
SVM: 0.5907881584006152
Random Forest: 0.8934763817629824
KNN: 0.717240223836876

Recall:
SVM: 0.7686274509803922
Random Forest: 0.8901960784313725
KNN: 0.7411764705882353

F1-score:
SVM: 0.6680753010738664
Random Forest: 0.8912039114156413
KNN: 0.7200654913227593
```

- For X_unfe we get least scores, where as for X we get high scores because it was standardised
- For X_transformed the scores are the highest because we had taken the most import features axis using PCA

Part 7-

- After PCA the correlation of features for starting has increased as evident by heatmap





Bonus-

- Because we consider Naive_Bayes we consider that Covariance matrix = 0 for(i not equal to j)
- EigenValues are the std , so we can directly take PCA descending_sorted wrt values of std

```
Accuracy:
SVM: 0.7686274509803922
Random Forest: 0.8627450980392157
KNN: 0.796078431372549

Precision:
SVM: 0.5907881584006152
Random Forest: 0.8536705503457421
KNN: 0.7705403572861749

Recall:
SVM: 0.7686274509803922
Random Forest: 0.8627450980392157
KNN: 0.796078431372549

F1-score:
SVM: 0.6680753010738664
Random Forest: 0.8561605179308049
KNN: 0.7719292547937051
```

- We can see the scores has decreased this is because now we neglected covariance between features.

Question 2:

Initials-

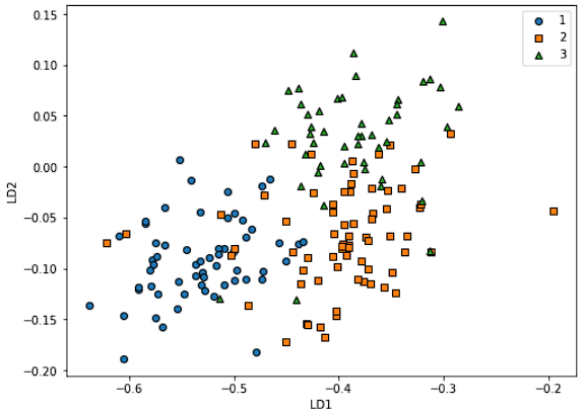
- Downloaded dataset using **wget** command called using **os.system**
- Load the **wine.data** dataset in df variable using **pd.read_csv**
- We hardcoded the columns names as it was not given in the dataset itself with the name
  features = ['Type of Wine','Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Total phenols', 'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins', 'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline']
- Used **df.describe()**
- Converted **df to X,y**
- Applied **MinMaxScaler()** to the data

Part 1-

- Implemented LDA_scratch with following functions
- **Fit, calculate_scatter_matrices, auto_select_var, transform, predict, score, calculate_class_prob**
- Also made a **scratch_cross_val_score** function

Part 2-

```
Selected_Feature From Most important to least important oredering
Malic acid
Ash
Magnesium
Total phenols
Hue
OD280/OD315 of diluted wines
Proline
Proanthocyanins
Color intensity
Flavanoids
Nonflavanoid phenols
Alcalinity of ash
Alcohol
```



Part 3-

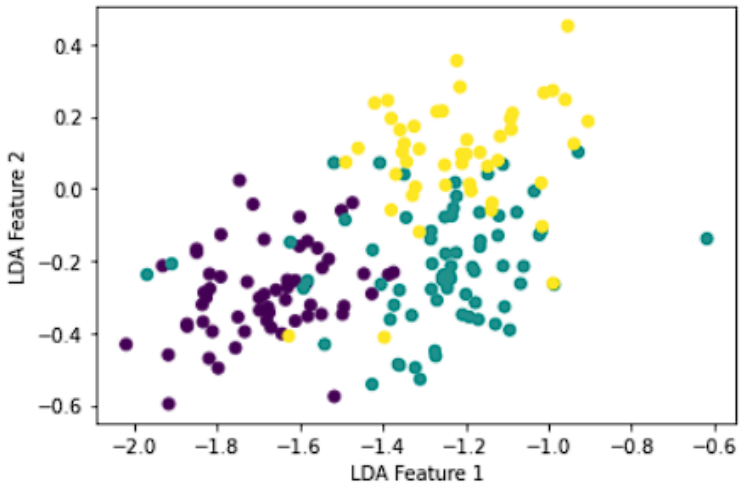
- Used KNN, RandomForest on LDA_scratch,PCA transformed X

```
PCA - KNN accuracy: 0.8611111111111112
PCA - Random Forest accuracy: 0.9444444444444444
LDA - KNN accuracy: 0.9166666666666666
LDA - Random Forest accuracy: 0.9722222222222222
```

- LDA is performing better here because LDA takes account for segregating data based on position where as PCA just minimises Covariance between features

Part 4-

Reduction Technique	Classifier	Accuracy
PCA	KNN	0.8611111111111112
PCA	Random Forest	0.9444444444444444
LDA	KNN	0.9166666666666666
LDA	Random Forest	0.9722222222222222



- Our LDA_scratch sorts the X_transformed in order of the std for each new feature, so we directly selected the first and 2nd feature to plot the above figure

Part 5-

- Using LDA_scratch fitted the X,y
- Then using scratch_cross_val_score found cv score
- `cv5=0.4057142857142857`
- As we have total 3 classes, so I took 3C2 class pairs then trained on LDA_scratch then plotted and computed ROC and AUC resepectively