*1a. Use Iris data from sklearn/scikit datasets or download it from the UCI ML repository:   
   
https://archive.ics.uci.edu/ml/machine-learning-databases/iris/   
   
Write a python program for Iris data visualization that implements the following:   
   
i) Correlation coefficient – display correlation matrix as heat maps (as in demo)*

Table

Description automatically generated

*ii) Feature distribution analysis (plot of individual features like in the demo) – display   
all the color coded features. (10 points; 5 pts/category data visualization plots)   
\*Note only provide data visualizations here*

A picture containing shoji, building, crossword puzzle, silhouette

Description automatically generated

*1b. Present your data analysis based on the following:   
   
a) implications of data distribution on data analysis*

In the figure, the plots in the diagonal represent the plot of the iris-features with respect to the species. The more the plots are overlapping with other features, the worse classifier that feature can be. For example: for the plot for sepal width distribution, we can see that all the three bells are overlapped to greater extent. The better a feature is for discriminating, the more separation there is in the distribution of classes for that feature. For example: the feature petal length can be used for discriminating the species as bell curves for the species are somehow separate. However, there is still some overlap between the species '1' and '2'. There is still a chance that the classifier will make some error in spite of using the feature 'petal length' but out of different features available, petal length can be considered a better feature for discriminating.

*b) inferences that can be drawn from i), ii) in 1a - what were the data   
patterns/trends observed and how do they influence data analysis. (5 points)   
\* Use data visualization to draw analysis*

**i. Correlation:**

The matrix in the correlation plot shows that the elements in the diagonals are always 1 and the elements are same in the upper triangle and in lower triangle.

The features that were found to be positively and strongly correlated are:

\* species and petal length

\*. sepal length and petal length

\* petal length and petal width

Moderate and positive correlation is found between:

\* sepal length and petal width

Similarly, the negatively correlated features are:

Weak and negatively correlated features:

\* sepal width and sepal length

\* sepal width and petal length

When two features are highly correlated, it means that they give similar insights into the target variable and that it might not be required to include both in a predictive model. The petal length and petal breadth in the iris dataset in this case are highly connected, and either one may be used to infer the species of the iris flower. To find hidden patterns or relationships that are not obvious when utilizing only one of the features, the combination of the two features may offer more in-depth information. In case of higly correlated features, in order to minimize the redundancy of model, these features selection or dimensionality reduction strategy might be required.

Likewise, weak correlations between features could mean that they are redundant or only somewhat useful with respect to the target variable.

**ii. Feature Distribution Analysis:**

Pair plots can be used to determine which features can be combined to improve class label discrimination. We can see that the stronger the correlation, the combination of those features is effective for classification. In the plot, we can see that the data points are more randomly distributed when sepal length and sepal width, for example, have a poor correlation (-0.12) between them. The class data points are not clearly separated. Yet, there is a definite distinction between the groups if we look at the petal length and petal width plot (which has a strong correlation coefficient 0.96).

*2. Use Iris dataset for a Linear Regression (LR) analysis using sklearn function in   
python. Drop the ‘petal length’ feature and train the LR model on:   
 i) 30% samples (i.e. train size = 0.3)*

Case: 30% training sample:

RMSE: 0.3342156170699589

Intercept: 0.8557618168628331

Coefficient: 0.62339783, -0.80150855, 1.11422801, 0.34931187

*ii) 70% samples (i.e. train size = 0.7)*

*Compare the LR parameters and perform quantitative performance analysis using the root   
mean squared error (RMSE) measure obtained in each case. Which case was better and   
why? Draw your analysis based on the evaluated parameters. (20 pts; 10pts/category)*

Case: 70% training example:

RMSE: 0.3245549572211216

Intercept: - 0.2725185759273119

coefficient: 0.67760814, -0.53212752, 1.00051066, 0.50420577

The RMSE for 30% training size is found to be higher than with 70% training sample. This might be because the higher sample size was better able to fit the model. The reason for higher RMSE for 30% training size is that it got fewer samples to learn from it.

*2b. Predict the ‘petal length’ for a sample X (e.g. sample index=100, or whichever   
sample that is not part of the training set) for both cases i) and ii). Compare the LR   
predictions with the actual value and evaluate the RMSE of each prediction. (10 pts)   
   
\*Note only provide the actual petal length and predicted petal values, LR and RMSE parameters here*For index=136

We have for petal length,

Actual value=5.6

For 30% train size,

predicted value=5.4308100

RMSE= 0.16918995114264668 (for predicted sample)

Intercept: 0.8557618168628331

Coefficient: 0.62339783, -0.80150855, 1.11422801, 0.34931187

For 70% train size,

predicted value=5.59681628

RMSE= 0.0031837190405461513 (for predicted sample)

Intercept: - 0.2725185759273119

coefficient: 0.67760814, -0.53212752, 1.00051066, 0.50420577

We can see that the model with train size=0.7 was able to perform better prediction than the model with train size of 0.3. This fact can be seen with lower RMSE for higher train size. This is because the model was able to learn from higher sample size than with the lower sample size. Due to this, it can better find pattern from the data in comparison with the lower train size and hence better predictions in unseen data.

APPENDIX:

from sklearn.datasets import load\_iris

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

#loading iris data

iris=load\_iris()

#creating pd dataframes

iris\_df=pd.DataFrame(data=iris.data,columns=iris.feature\_names)

target\_df=pd.DataFrame(data=iris.target,columns=['species'])

#concatenating the dataframes:

df=pd.concat([iris\_df,target\_df],axis=1)

#visualize iris features as a heatmap

cor\_eff=df.corr()

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

sns.heatmap(cor\_eff,linecolor='white',linewidths=1,cmap='Pastel2', annot=True)

#iris feature analysis:

g=sns.pairplot(df,hue='species')

data = pd.DataFrame(load\_iris().data,columns=["sepal length","sepal width","petal length","petal width"])

y = pd.DataFrame(load\_iris().target,columns=["species"])

df= pd.concat([data,y],axis=1)

X=df.drop(columns=['petal length'])

Y=df['petal length']

def train(train\_size1,train\_size2):

print("==============================================================")

print(f"for training size of {train\_size1\*100}% :")

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,train\_size=train\_size1,random\_state=111)

X\_train2,X\_test2,Y\_train2,Y\_test2=train\_test\_split(X,Y,train\_size=train\_size2,random\_state=111)

lr=LinearRegression()

lr.fit(X\_train,Y\_train)

Y\_pred=lr.predict(X\_test)

rmse = np.sqrt(mean\_squared\_error(Y\_test, Y\_pred))

print("RMSE is :",rmse)

print("Intercept is : ",lr.intercept\_)

print("coefficient is: ",lr.coef\_)

#create new data point for prediction

index = set(X\_test2.index.tolist()).intersection(X\_test.index.tolist())

test\_X = pd.DataFrame(X, index=[136])

test\_y = pd.DataFrame(Y, index=[136])

predicted\_y = lr.predict(test\_X)

print("Predicted value:",predicted\_y[0])

print("Actual Value: ",test\_y.iloc[0]['petal length'])

print(f"Rmse for prediction of train size {train\_size1} is",np.sqrt(mean\_squared\_error(test\_y, predicted\_y)))

print(("=============================================================="))

print(f"for training size of {train\_size2\*100}% :")

lr2=LinearRegression()

lr2.fit(X\_train2,Y\_train2)

Y\_pred2=lr.predict(X\_test2)

rmse2 = np.sqrt(mean\_squared\_error(Y\_test2, Y\_pred2))

predicted\_y2 = lr2.predict(test\_X)

print("RMSE is :",rmse2)

print("Intercept is : ",lr2.intercept\_)

print("coefficient is: ",lr2.coef\_)

print("Predicted value:",predicted\_y2)

print("Actual Value: ",test\_y.iloc[0]["petal length"])

print(f"Rmse for prediction of train size {train\_size2} is",np.sqrt(mean\_squared\_error(test\_y, predicted\_y2)))

print(("=============================================================="))

train(0.3,0.7)