**Question 1**

1. Implement KNN classification in iris data set  
2. Choose different values of k (from 1-25)

3. Provide the confusion Matrix for same.  
4. Then implement logistics regression for Iris data  
5. Finally compare kNN and Logistic Regression

**Source Code**

*import numpy as np*

*import matplotlib.pyplot as plt*

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

*from sklearn.neighbors import KNeighborsClassifier*

*from sklearn.linear\_model import LogisticRegression*

*from sklearn.datasets import load\_iris*

*from sklearn.metrics import confusion\_matrix, accuracy\_score*

*iris = load\_iris()*

*X = iris.data*

*y = iris.target*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)*

*# KNN Classifier with k=3*

*knn\_3 = KNeighborsClassifier(n\_neighbors=3)*

*knn\_3.fit(X\_train, y\_train)*

*y\_pred\_knn\_3 = knn\_3.predict(X\_test)*

*conf\_matrix\_knn\_3 = confusion\_matrix(y\_test, y\_pred\_knn\_3)*

*accuracy\_knn\_3 = accuracy\_score(y\_test, y\_pred\_knn\_3)*

*# KNN Classifier with k=13*

*knn\_13 = KNeighborsClassifier(n\_neighbors=13)*

*knn\_13.fit(X\_train, y\_train)*

*y\_pred\_knn\_13 = knn\_13.predict(X\_test)*

*conf\_matrix\_knn\_13 = confusion\_matrix(y\_test, y\_pred\_knn\_13)*

*accuracy\_knn\_13 = accuracy\_score(y\_test, y\_pred\_knn\_13)*

*# KNN Classifier with k=20*

*knn\_20 = KNeighborsClassifier(n\_neighbors=20)*

*knn\_20.fit(X\_train, y\_train)*

*y\_pred\_knn\_20 = knn\_20.predict(X\_test)*

*conf\_matrix\_knn\_20 = confusion\_matrix(y\_test, y\_pred\_knn\_20)*

*accuracy\_knn\_20 = accuracy\_score(y\_test, y\_pred\_knn\_20)*

*# Logistic Regressor*

*log\_reg = LogisticRegression(max\_iter=200)*

*log\_reg.fit(X\_train, y\_train)*

*y\_pred\_log\_reg = log\_reg.predict(X\_test)*

*conf\_matrix\_log\_reg = confusion\_matrix(y\_test, y\_pred\_log\_reg)*

*accuracy\_log\_reg = accuracy\_score(y\_test, y\_pred\_log\_reg)*

*print(f'Accuracy of KNN (k=3): {accuracy\_knn\_3 \* 100:.2f}%')*

*print(f'Accuracy of KNN (k=13): {accuracy\_knn\_13 \* 100:.2f}%')*

*print(f'Accuracy of KNN (k=20): {accuracy\_knn\_20 \* 100:.2f}%')*

*print(f'Accuracy of Logistic Regression: {accuracy\_log\_reg \* 100:.2f}%')*

*fig, axes = plt.subplots(2, 2, figsize=(8, 8))*

*# Plot for KNN (k=3)*

*axes[0][0].imshow(conf\_matrix\_knn\_3, interpolation='nearest', cmap=plt.cm.Blues)*

*axes[0][0].set\_title('Confusion Matrix (KNN k=3)')*

*axes[0][0].set\_xticks(np.arange(3))*

*axes[0][0].set\_yticks(np.arange(3))*

*axes[0][0].set\_xticklabels(iris.target\_names)*

*axes[0][0].set\_yticklabels(iris.target\_names)*

*axes[0][0].set\_xlabel('Predicted Label')*

*axes[0][0].set\_ylabel('True Label')*

*for i in range(3):*

*for j in range(3):*

*axes[0][0].text(j, i, conf\_matrix\_knn\_3[i, j], ha="center", va="center", color="white")*

*# Plot for KNN (k=13)*

*axes[0][1].imshow(conf\_matrix\_knn\_13, interpolation='nearest', cmap=plt.cm.Blues)*

*axes[0][1].set\_title('Confusion Matrix (KNN k=13)')*

*axes[0][1].set\_xticks(np.arange(3))*

*axes[0][1].set\_yticks(np.arange(3))*

*axes[0][1].set\_xticklabels(iris.target\_names)*

*axes[0][1].set\_yticklabels(iris.target\_names)*

*axes[0][1].set\_xlabel('Predicted Label')*

*axes[0][1].set\_ylabel('True Label')*

*for i in range(3):*

*for j in range(3):*

*axes[0][1].text(j, i, conf\_matrix\_knn\_13[i, j], ha="center", va="center", color="white")*

*# Plot for KNN (k=20)*

*axes[1][0].imshow(conf\_matrix\_knn\_20, interpolation='nearest', cmap=plt.cm.Blues)*

*axes[1][0].set\_title('Confusion Matrix (KNN k=20)')*

*axes[1][0].set\_xticks(np.arange(3))*

*axes[1][0].set\_yticks(np.arange(3))*

*axes[1][0].set\_xticklabels(iris.target\_names)*

*axes[1][0].set\_yticklabels(iris.target\_names)*

*axes[1][0].set\_xlabel('Predicted Label')*

*axes[1][0].set\_ylabel('True Label')*

*for i in range(3):*

*for j in range(3):*

*axes[1][0].text(j, i, conf\_matrix\_knn\_20[i, j], ha="center", va="center", color="white")*

*# Plot for Logistic Regression*

*axes[1][1].imshow(conf\_matrix\_log\_reg, interpolation='nearest', cmap=plt.cm.Blues)*

*axes[1][1].set\_title('Confusion Matrix (Logistic Regression)')*

*axes[1][1].set\_xticks(np.arange(3))*

*axes[1][1].set\_yticks(np.arange(3))*

*axes[1][1].set\_xticklabels(iris.target\_names)*

*axes[1][1].set\_yticklabels(iris.target\_names)*

*axes[1][1].set\_xlabel('Predicted Label')*

*axes[1][1].set\_ylabel('True Label')*

*for i in range(3):*

*for j in range(3):*

*axes[1][1].text(j, i, conf\_matrix\_log\_reg[i, j], ha="center", va="center", color="white")*

*plt.tight\_layout()*

*plt.show()*

*accuracies = {*

*"KNN (k=3)": accuracy\_knn\_3,*

*"KNN (k=13)": accuracy\_knn\_13,*

*"KNN (k=20)": accuracy\_knn\_20,*

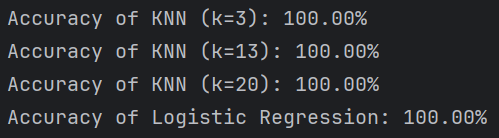
*"Logistic Regression": accuracy\_log\_reg*

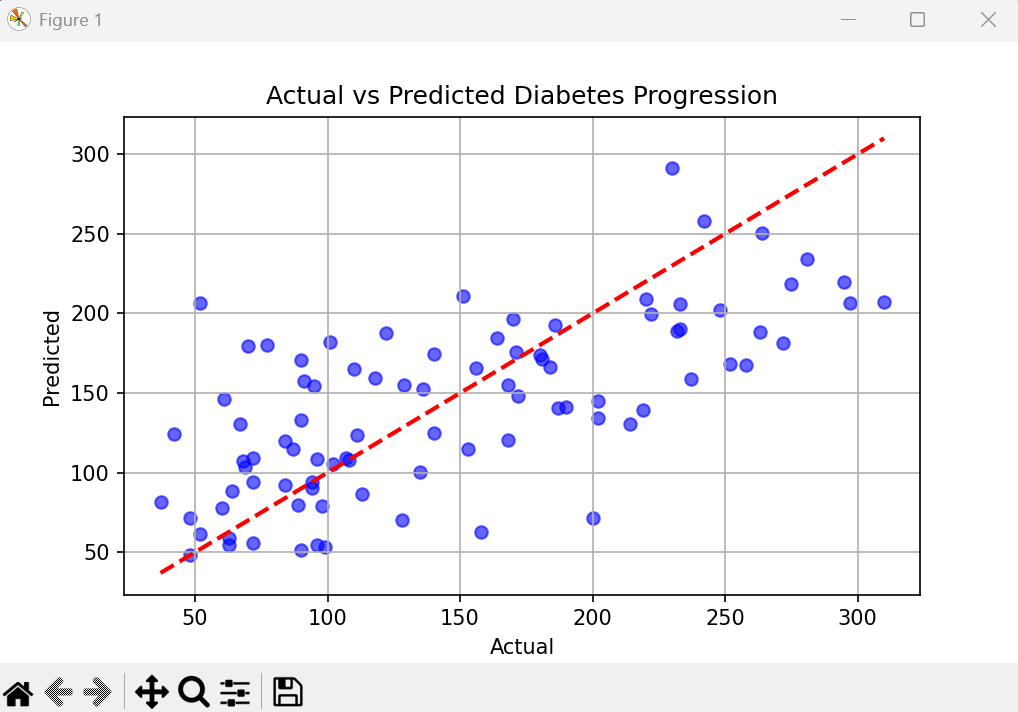
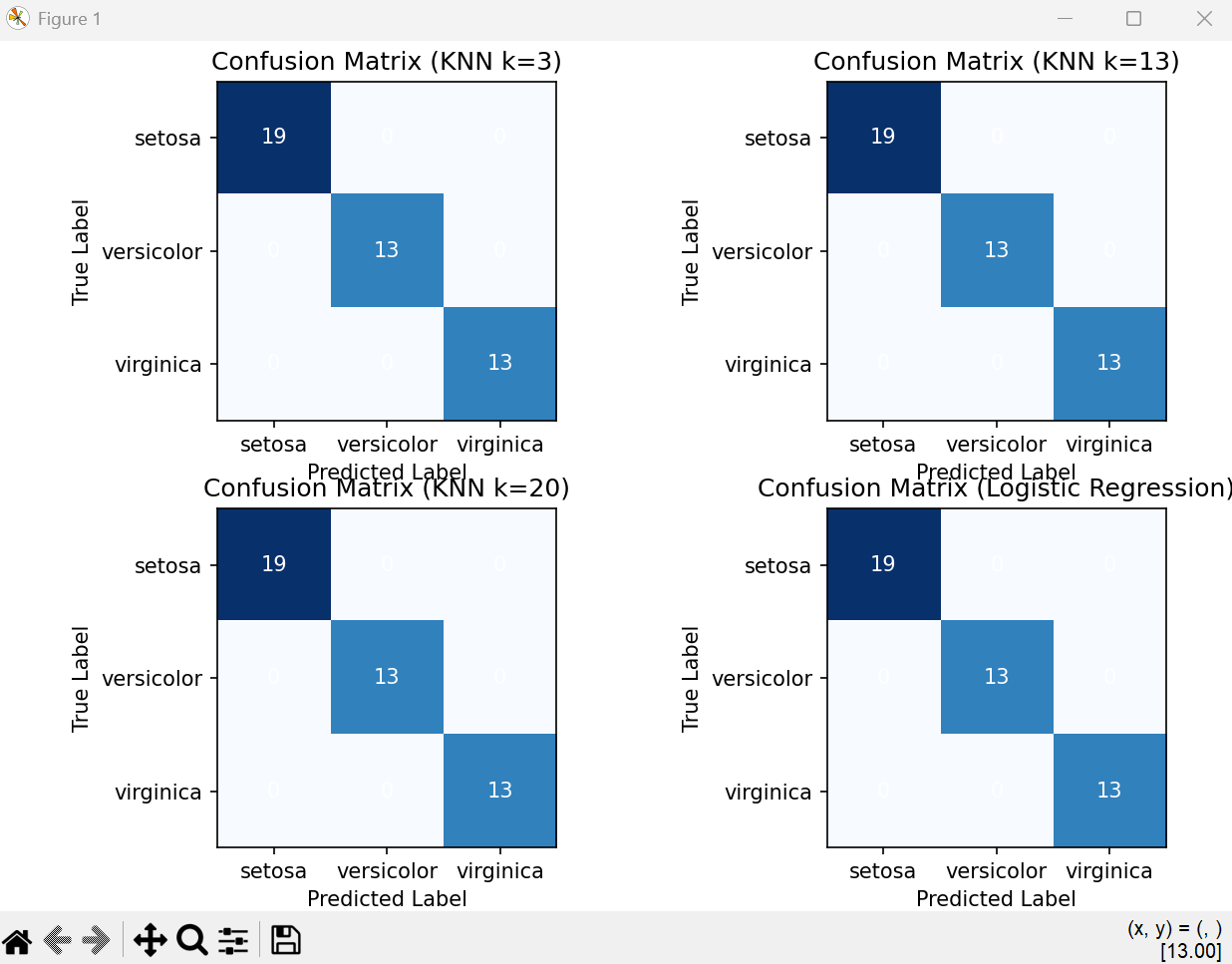
*}*

*best\_model = max(accuracies, key=accuracies.get)*

*print(f"\nThe best performing model is: {best\_model}.")*

**Output**

*Terminal*

*Visualization*