**Name:** Arham Sharif

**Seat No.:** EB21102022

**Section:** B

**Subject:** Data Warehouse & Data Mining

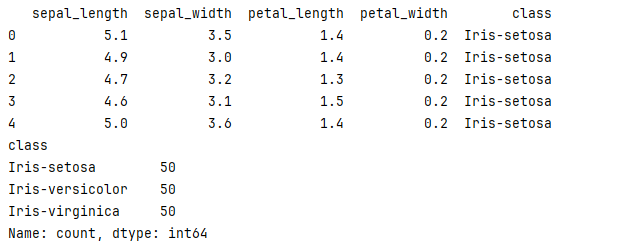
**CLASSIFICATION**

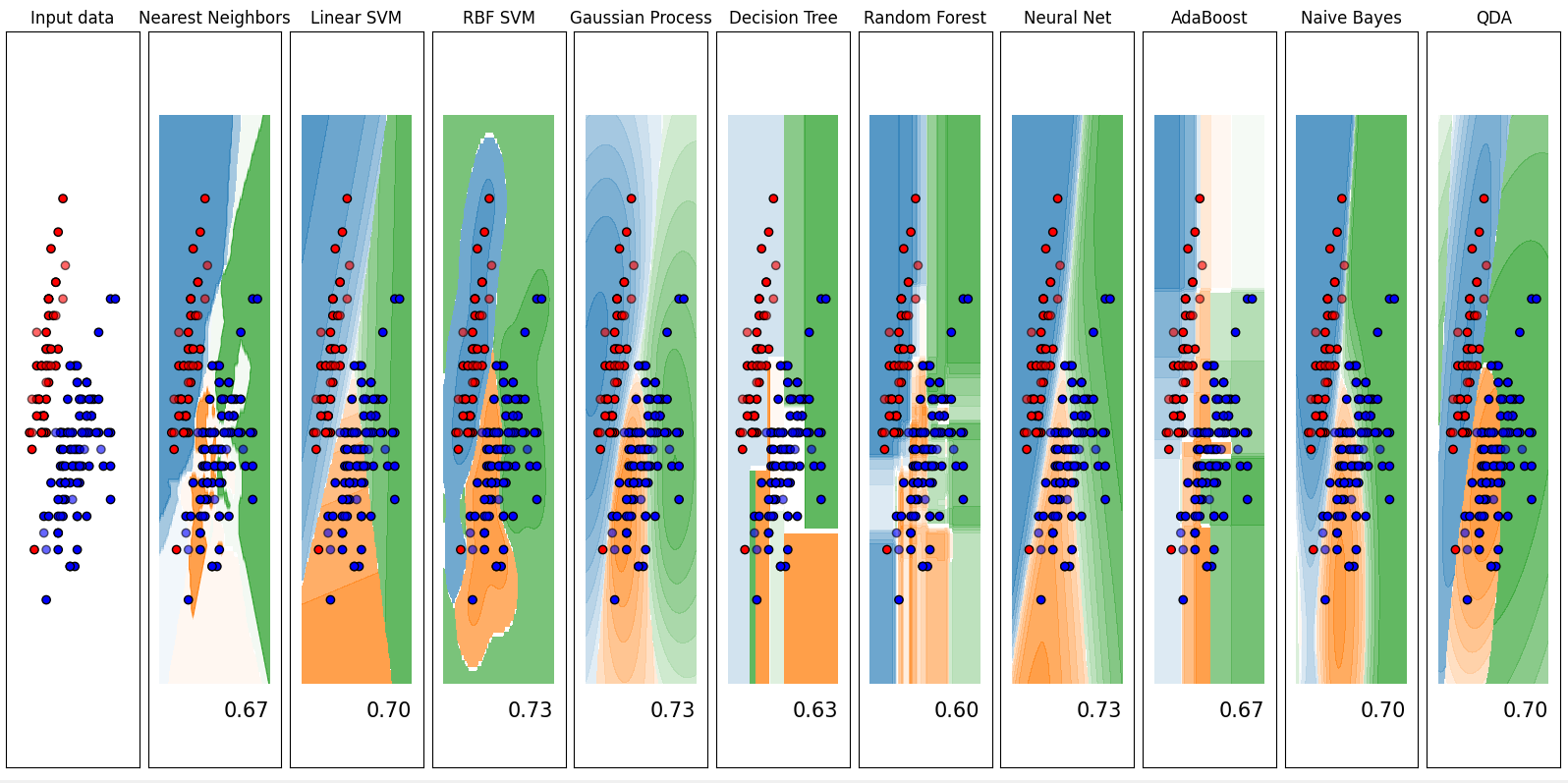
**IRIS – CLASSIFIER**

**CODE**

*# 1. Import Libraries*import pandas as pd  
import matplotlib.pyplot as plt  
from matplotlib.colors import ListedColormap  
  
from sklearn.preprocessing import LabelEncoder  
from sklearn.model\_selection import train\_test\_split  
from sklearn.pipeline import make\_pipeline  
from sklearn.preprocessing import StandardScaler  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.svm import SVC  
from sklearn.gaussian\_process import GaussianProcessClassifier  
from sklearn.gaussian\_process.kernels import RBF  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier  
from sklearn.neural\_network import MLPClassifier  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.discriminant\_analysis import QuadraticDiscriminantAnalysis  
from sklearn.inspection import DecisionBoundaryDisplay  
  
*# 2. Import Data from online URL*url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"  
  
*# The iris data doesn't have headers, so add them manually*column\_names = ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'class']  
df = pd.read\_csv(url, header=None, names=column\_names)  
  
*# 3. Preprocess / EDA / IDA*print(df.head())  
print(df['class'].value\_counts())  
  
*# Use only two features for visualization*X = df[['sepal\_length', 'sepal\_width']].values  
y = df['class'].values  
  
*# Encode target labels*le = LabelEncoder()  
y = le.fit\_transform(y)  
  
*# 4. Train-Test Split (80% Train / 20% Test)*X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.2, random\_state=42, stratify=y  
)  
  
*# 5. Execute / Feature Scaling inside pipeline  
  
# 6. Define classifiers*names = [  
 "Nearest Neighbors",  
 "Linear SVM",  
 "RBF SVM",  
 "Gaussian Process",  
 "Decision Tree",  
 "Random Forest",  
 "Neural Net",  
 "AdaBoost",  
 "Naive Bayes",  
 "QDA",  
]  
  
classifiers = [  
 KNeighborsClassifier(3),  
 SVC(kernel="linear", C=0.025, random\_state=42),  
 SVC(gamma=2, C=1, random\_state=42),  
 GaussianProcessClassifier(1.0 \* RBF(1.0), random\_state=42),  
 DecisionTreeClassifier(max\_depth=5, random\_state=42),  
 RandomForestClassifier(max\_depth=5, n\_estimators=10, max\_features=1, random\_state=42),  
 MLPClassifier(alpha=1, max\_iter=1000, random\_state=42),  
 AdaBoostClassifier(random\_state=42),  
 GaussianNB(),  
 QuadraticDiscriminantAnalysis(),  
]  
  
*# 7. Testing and 8. Predict + Visualize decision boundaries*figure = plt.figure(figsize=(27, 9))  
cm\_bright = ListedColormap(["#FF0000", "#0000FF"])  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
  
*# Plot input data*ax = plt.subplot(1, len(classifiers) + 1, 1)  
ax.set\_title("Input data")  
ax.scatter(X\_train[:, 0], X\_train[:, 1], c=y\_train, cmap=cm\_bright, edgecolors="k")  
ax.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_test, cmap=cm\_bright, alpha=0.6, edgecolors="k")  
ax.set\_xlim(x\_min, x\_max)  
ax.set\_ylim(y\_min, y\_max)  
ax.set\_xticks(())  
ax.set\_yticks(())  
  
for i, (name, clf) in enumerate(zip(names, classifiers), start=2):  
 ax = plt.subplot(1, len(classifiers) + 1, i)  
 clf\_pipeline = make\_pipeline(StandardScaler(), clf)  
 clf\_pipeline.fit(X\_train, y\_train)  
 score = clf\_pipeline.score(X\_test, y\_test)  
  
 DecisionBoundaryDisplay.from\_estimator(  
 clf\_pipeline, X, alpha=0.8, ax=ax, eps=0.5  
 )  
  
 ax.scatter(X\_train[:, 0], X\_train[:, 1], c=y\_train, cmap=cm\_bright, edgecolors="k")  
 ax.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_test, cmap=cm\_bright, alpha=0.6, edgecolors="k")  
 ax.set\_xlim(x\_min, x\_max)  
 ax.set\_ylim(y\_min, y\_max)  
 ax.set\_xticks(())  
 ax.set\_yticks(())  
 ax.set\_title(name)  
  
 ax.text(  
 x\_max - 0.5,  
 y\_min + 0.3,  
 f"{score:.2f}",  
 size=15,  
 horizontalalignment="right",  
 )  
  
plt.tight\_layout()  
plt.show()

**OUTPUT**

****

****

**RECOGNIZING HAND-WRITTEN DIGITS – CLASSIFIER**

**CODE**

*# 1. Import Libraries*import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.pipeline import make\_pipeline  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.svm import SVC  
from sklearn.metrics import classification\_report, accuracy\_score  
from sklearn.inspection import DecisionBoundaryDisplay  
from matplotlib.colors import ListedColormap  
  
*# 2. Load Digits Dataset*digits = datasets.load\_digits()  
X = digits.data  
y = digits.target  
  
print(f"Dataset shape: {X.shape}")  
print(f"Number of classes: {len(np.unique(y))}")  
  
*# 3. Train-Test Split (80% train, 20% test)*X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.2, random\_state=42, stratify=y  
)  
  
*# 4. Define classifier pipeline (example: KNN with scaling)*clf = make\_pipeline(StandardScaler(), KNeighborsClassifier(n\_neighbors=3))  
  
*# 5. Train classifier*clf.fit(X\_train, y\_train)  
  
*# 6. Test classifier*y\_pred = clf.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))  
print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}")  
  
*# 7. Visualize some predictions*fig, axes = plt.subplots(2, 5, figsize=(10, 5))  
for ax, image, pred, true in zip(axes.flatten(), X\_test, y\_pred, y\_test):  
 ax.imshow(image.reshape(8, 8), cmap='gray')  
 ax.set\_title(f"Pred: {pred}, True: {true}")  
 ax.axis('off')  
plt.tight\_layout()  
plt.show()

**OUTPUT**

