**KNN ALGORITHM**

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Load the dataset

url = r"C:\Users\HP\Desktop\project\voice.csv" # Replace with the actual path or URL to your dataset

df = pd.read\_csv(url)

# Separate features (X) and labels (y)

X = df.iloc[:, 1:-1] # Exclude the 'label' column

y = df['label']

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Test different values of k

k\_values = [10,20,30,40,50,60,70,80,90] # You can adjust the list of k values to test

accuracies = []

for k in k\_values:

# Initialize k-NN classifier

knn\_classifier = KNeighborsClassifier(n\_neighbors=k)

# Train the classifier

knn\_classifier.fit(X\_train\_scaled, y\_train)

# Make predictions on the test set

y\_pred = knn\_classifier.predict(X\_test\_scaled)

# Evaluate the accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

accuracies.append(accuracy)

# Print the accuracy for each k

print(f"Accuracy for k={k}: {accuracy \* 100:.2f}%")

# Plot the line graph

plt.plot(k\_values, accuracies, marker='o')

plt.title('Accuracy vs. k Value for k-NN Classifier')

plt.xlabel('k Value')

plt.ylabel('Accuracy')

plt.grid(True)

plt.show()

Accuracy for k=10: 97.95%

Accuracy for k=20: 97.48%

Accuracy for k=30: 96.69%

Accuracy for k=40: 96.53%

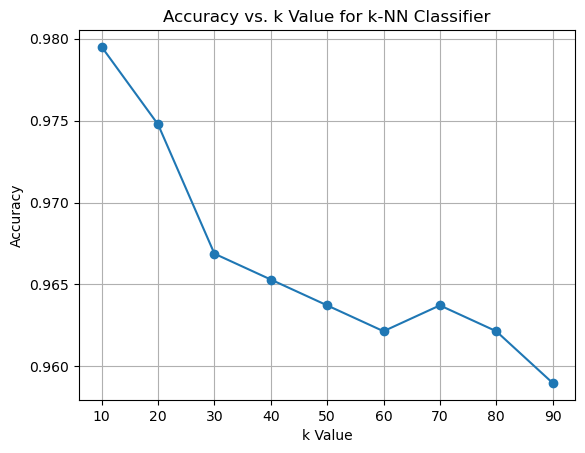
Accuracy for k=50: 96.37%

Accuracy for k=60: 96.21%

Accuracy for k=70: 96.37%

Accuracy for k=80: 96.21%

Accuracy for k=90: 95.90%



**RANDOM FOREST**

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

# Load your dataset

# Replace 'your\_dataset.csv' with the actual file path

data = pd.read\_csv(r"C:\Users\HP\Desktop\project\voice.csv")

# Separate features (X) and labels (y)

X = data.drop('label', axis=1) # Assuming 'label' is the column containing gender labels

y = data['label']

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Test different values of n\_estimators

n\_estimators\_values = [10,20,30,40,50,60,70,80,90] # You can adjust the list of n\_estimators values to test

accuracies = []

for n\_estimators in n\_estimators\_values:

# Initialize the Random Forest classifier

rf\_classifier = RandomForestClassifier(n\_estimators=n\_estimators, random\_state=42)

# Train the classifier

rf\_classifier.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = rf\_classifier.predict(X\_test)

# Evaluate the accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

accuracies.append(accuracy)

# Print the accuracy for each n\_estimators

print(f"Accuracy for n\_estimators={n\_estimators}: {accuracy \* 100:.2f}%")

# Plot the line graph

plt.plot(n\_estimators\_values, accuracies, marker='o')

plt.title('Accuracy vs. Number of Estimators for Random Forest Classifier')

plt.xlabel('Number of Estimators')

plt.ylabel('Accuracy')

plt.grid(True)

plt.show()

Accuracy for n\_estimators=10: 97.79%

Accuracy for n\_estimators=20: 97.63%

Accuracy for n\_estimators=30: 98.42%

Accuracy for n\_estimators=40: 98.11%

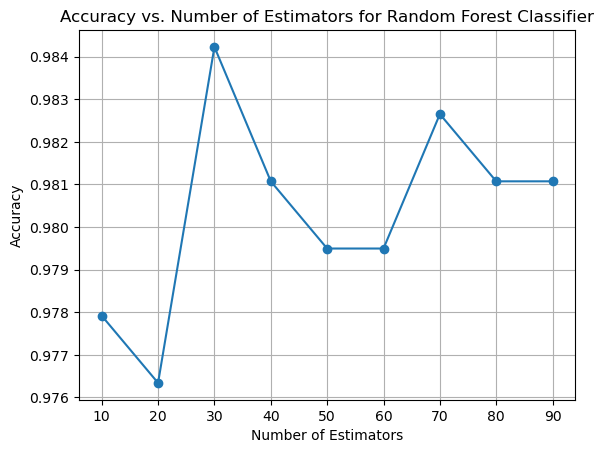
Accuracy for n\_estimators=50: 97.95%

Accuracy for n\_estimators=60: 97.95%

Accuracy for n\_estimators=70: 98.26%

Accuracy for n\_estimators=80: 98.11%

Accuracy for n\_estimators=90: 98.11%



**SVM ALGORITHM**

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler

# Load the dataset (replace 'your\_dataset.csv' with the actual file name)

df = pd.read\_csv(r"C:\Users\HP\Desktop\project\voice.csv")

# Separate features and labels

X = df.iloc[:, 1:-1] # Assuming the features are in columns 1 to second-to-last

y = df['label'] # Assuming the labels are in the last column

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize the features (optional but recommended for SVM)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Test different values of C

C\_values = [10, 20, 30, 40,50,60,70,80,90] # You can adjust the list of C values to test

accuracies = []

for C in C\_values:

# Create an SVM classifier with a linear kernel

svm\_classifier = SVC(kernel='linear', C=C)

# Train the SVM classifier

svm\_classifier.fit(X\_train\_scaled, y\_train)

# Make predictions on the test set

y\_pred = svm\_classifier.predict(X\_test\_scaled)

# Evaluate the accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

accuracies.append(accuracy)

# Print the accuracy for each C

print(f"Accuracy for C={C}: {accuracy \* 100:.2f}%")

# Plot the line graph

plt.plot(C\_values, accuracies, marker='o')

plt.title('Accuracy vs. C Value for SVM Classifier')

plt.xlabel('C Value')

plt.ylabel('Accuracy')

plt.grid(True)

plt.show()

Accuracy for C=10: 97.63%

Accuracy for C=20: 97.63%

Accuracy for C=30: 97.48%

Accuracy for C=40: 97.48%

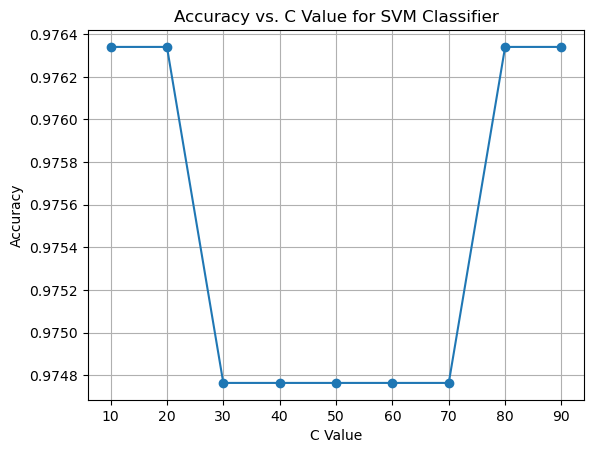
Accuracy for C=50: 97.48%

Accuracy for C=60: 97.48%

Accuracy for C=70: 97.48%

Accuracy for C=80: 97.63%

Accuracy for C=90: 97.63%



**LR ALGORITHM**

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

# Load the dataset

data = pd.read\_csv(r"C:\Users\HP\Desktop\project\voice.csv") # Replace 'your\_dataset.csv' with the actual file path

# Separate features (X) and labels (y)

X = data.drop('label', axis=1) # Assuming 'label' is the column containing gender labels

y = data['label']

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize the features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Test different values of C

C\_values = [10,20,30,40,50,60,70,80,90] # You can adjust the list of C values to test

accuracies = []

for C in C\_values:

# Initialize logistic regression model

model = LogisticRegression(C=C)

# Train the model

model.fit(X\_train\_scaled, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test\_scaled)

# Evaluate the accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

accuracies.append(accuracy)

# Print the accuracy for each C

print(f"Accuracy for C={C}: {accuracy \* 100:.2f}%")

# Plot the line graph

plt.plot(C\_values, accuracies, marker='o')

plt.title('Accuracy vs. C Value for Logistic Regression')

plt.xlabel('C Value')

plt.ylabel('Accuracy')

plt.xscale('log') # Use a log scale for better visualization if C values vary widely

plt.grid(True)

plt.show()

Accuracy for C=10: 98.11%

Accuracy for C=20: 98.11%

Accuracy for C=30: 98.11%

Accuracy for C=40: 98.11%

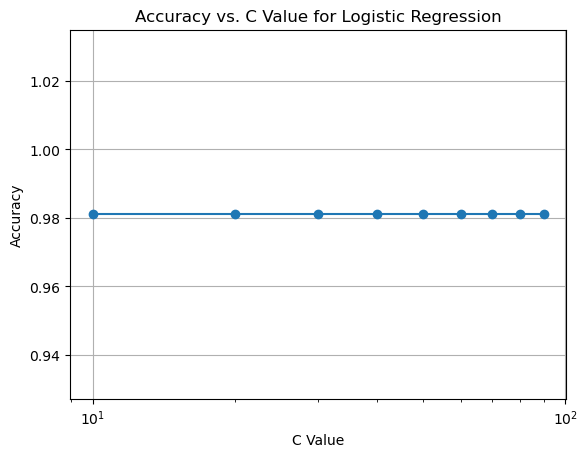
Accuracy for C=50: 98.11%

Accuracy for C=60: 98.11%

Accuracy for C=70: 98.11%

Accuracy for C=80: 98.11%

Accuracy for C=90: 98.11%



**NB ALGORITHM**

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

# Load your dataset

df = pd.read\_csv(r"C:\Users\HP\Desktop\project\voice.csv")

# Select features (X) and target variable (y)

X = df.drop('label', axis=1) # Features

y = df['label'] # Target variable

# Test different values of test\_size

test\_sizes = [10,20,30,40,50,60,70,80,90] # You can adjust the list of test sizes to test

accuracies = []

for test\_size in test\_sizes:

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize Gaussian Naive Bayes classifier

model = GaussianNB()

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

accuracies.append(accuracy)

# Print the accuracy for each test size

print(f"Accuracy for test\_size={test\_size}: {accuracy \* 100:.2f}%")

# Plot the line graph

plt.plot(test\_sizes, accuracies, marker='o')

plt.title('Accuracy vs. Test Size for Gaussian Naive Bayes')

plt.xlabel('Test Size')

plt.ylabel('Accuracy')

plt.grid(True)

plt.show()

Accuracy for C=10: 98.11%

Accuracy for C=20: 98.11%

Accuracy for C=30: 98.11%

Accuracy for C=40: 98.11%

Accuracy for C=50: 98.11%

Accuracy for C=60: 98.11%

Accuracy for C=70: 98.11%

Accuracy for C=80: 98.11%

Accuracy for C=90: 98.11%

