```
data=pd.read_csv("/content/drive/MyDrive/dataset/processed_thyroid_data.csv")
data

data = data.dropna()
data.tail()

data.isnull().sum()
standardScaler = StandardScaler()
columns_to_scale = ['age', 'TT4', 'TSH', 'T3','T4U','FTI']
data[columns_to_scale] = standardScaler.fit_transform(data[columns_to_scale])



y = data["binaryClass"]
X = data.drop('binaryClass', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=5)

from sklearn.ensemble import RandomForestClassifier
model1=RandomForestClassifier()
model1.fit(X_train,y_train)

RandomForestClassifier()
y_predict1=model1.predict(X_test)
v=confusion_matrix(y_test,y_predict1)
cm=ConfusionMatrixDisplay(confusion_matrix=v)
cm.plot()

accuracy = accuracy_score(y_test,y_predict1)
print(f'Accuracy: {accuracy * 100:.2f}%')
precision = precision_score(y_test,y_predict1)
print(f'Precision: {precision * 100:.2f}%')
recall=recall_score(y_test,y_predict1)
```

```
print(f'Recall: {recall * 100:.2f}%')
f1 = f1_score(y_test, y_predict1, average='binary')
print(f'F1 Score: {f1*100:.2f}')

tn, fp, fn, tp = confusion_matrix(y_test, y_predict1).ravel()

specificity = tn / (tn + fp)
print(f'Specificity: {specificity*100:.2f}')

fpr = fp / (fp + tn)
print(f'False Positive Rate: {fpr*100:.2f}')

fnr = fn / (fn + tp)
print(f'False Negative Rate: {fnr*100:.2f}')



from sklearn.svm import SVC

model2 = SVC(C=30 ,kernel= 'rbf')
In [14]:
model2.fit(X_train,y_train)
y_predict2=model2.predict(X_test)

v=confusion_matrix(y_test,y_predict2)
cm=ConfusionMatrixDisplay(confusion_matrix=v)
cm.plot()

model3=LogisticRegression(C= 20, penalty= 'l1', solver= 'liblinear')
model3.fit(X_train,y_train)
y_predict3=model3.predict(X_test)
```

```
v=confusion_matrix(y_test,y_predict3)
cm=ConfusionMatrixDisplay(confusion_matrix=v)
cm.plot()

accuracy = accuracy_score(y_test,y_predict3)
print(f'Accuracy: {accuracy * 100:.2f}%')
precision = precision_score(y_test,y_predict3)
print(f'Precision: {precision * 100:.2f}%')
recall=recall_score(y_test,y_predict3)
print(f'Recall: {recall * 100:.2f}%')
f1 = f1_score(y_test, y_predict3, average='binary')
print(f'F1 Score: {f1*100:.2f}')

tn, fp, fn, tp = confusion_matrix(y_test, y_predict3).ravel()

# Calculate specificity
specificity = tn / (tn + fp)
print(f'Specificity: {specificity*100:.2f}')

# Calculate false positive rate (FPR)
fpr = fp / (fp + tn)
print(f'False Positive Rate: {fpr*100:.2f}')

# Calculate false negative rate (FNR)
fnr = fn / (fn + tp)
print(f'False Negative Rate: {fnr*100:.2f}')

model4 = DecisionTreeClassifier()
model4.fit(X_train, y_train)

DecisionTreeClassifier()
```

```
y_predict4 = model4.predict(X_test)
accuracy = accuracy_score(y_test, y_predict4)
print(f'Accuracy: {accuracy * 100:.2f}%')
precision = precision_score(y_test,y_predict4)
print(f'Precision: {precision * 100:.2f}%')
recall=recall_score(y_test,y_predict4)
print(f'Recall: {recall * 100:.2f}%')
f1 = f1_score(y_test, y_predict4, average='binary')
print(f'F1 Score: {f1*100:.2f}')

tn, fp, fn, tp = confusion_matrix(y_test, y_predict4).ravel()

# Calculate specificity
specificity = tn / (tn + fp)
print(f'Specificity: {specificity*100:.2f}')

# Calculate false positive rate (FPR)
fpr = fp / (fp + tn)
print(f'False Positive Rate: {fpr*100:.2f}')

# Calculate false negative rate (FNR)
fnr = fn / (fn + tp)
print(f'False Negative Rate: {fnr*100:.2f}')

from sklearn.neighbors import KNeighborsClassifier
model5 = KNeighborsClassifier(n_neighbors=3)
model5.fit(X_train, y_train)

KNeighborsClassifier(n_neighbors=3)
y_predict5 = model5.predict(X_test)
accuracy = accuracy_score(y_test, y_predict5)
```

```
print(f'Accuracy: {accuracy * 100:.2f}%')
precision = precision_score(y_test,y_predict5)
print(f'Precision: {precision * 100:.2f}%')
recall=recall_score(y_test,y_predict5)
print(f'Recall: {recall * 100:.2f}%')
f1 = f1_score(y_test, y_predict5, average='binary')
print(f'F1 Score: {f1*100:.2f}')

tn, fp, fn, tp = confusion_matrix(y_test, y_predict5).ravel()

# Calculate specificity
specificity = tn / (tn + fp)
print(f'Specificity: {specificity*100:.2f}')

# Calculate false positive rate (FPR)
fpr = fp / (fp + tn)
print(f'False Positive Rate: {fpr*100:.2f}')

# Calculate false negative rate (FNR)
fnr = fn / (fn + tp)
print(f'False Negative Rate: {fnr*100:.2f}')

from sklearn.decomposition import PCA
labels_rfc = model1.predict(X_train)
labels_svm = model2.predict(X_train)
labels_lr = model3.predict(X_train)
labels_dtc = model4.predict(X_train)
labels_knn = model5.predict(X_train)

model_labels = [labels_rfc, labels_svm, labels_lr, labels_dtc, labels_knn]
```

```
model_names = ['RandomForestClassifier', 'SVM', 'Logistic Regression',  'Decision Tree Classifier','KNN']


plt.figure(figsize=(12, 8))

pca = PCA(n_components=28)

pca_result = pca.fit_transform(X_train)

markers = ['o', 's', 'D', '^', 'v']

colors = ['b', 'g', 'r', 'c', 'm']

for i, (name, labels, marker, color) in enumerate(zip(model_names, model_labels, markers, colors)):

  sns.scatterplot(x=pca_result[:, 0], y=pca_result[:, 1], hue=labels, palette='viridis', legend=None, alpha=0.6, marker=marker, edgecolor='w', label=name)

plt.title('Clustering Results of All Models')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.legend(title='Models')

plt.show()
```