15 DE MAYO DE 2024

PROYECTO

CRUZ GALLEGOS RAMSES AARÓN
MORALES ZEPEDA IVAN YUTLANIH
SEMINARIO DE SOL DE PROBLEMAS DE INTELIGENCIA ARTIFICIAL 2
D05
MTRO: CAMPOS PEÑA DIEGO

Índice

Introducción

Los clasificadores como Regresión Logística (Logistic Regression), K-Vecinos Cercanos (K-Nearest Neighbors), Máquinas de Vectores de Soporte (Support Vector Machines) y Naive Bayes son muy importantes al asignar categorías a diferentes instancias de datos.
Cada uno de estos métodos tiene sus propias características, ventajas y desventajas que los hacen adecuados para diferentes tipos de problemas de clasificación. Se analizará su desempeño utilizando un conjunto de datos (decidimos juntar los dataset que se nos proporcionaron en uno solo para tener un mejor control de los datos): zoo.cvs

Desarrollo

El código se estructuró en funciones para cargar los datos, entrenar los modelos y evaluar su desempeño. Primero, se carga el conjunto de datos utilizando la función load_data():

```
def load_data():
    # Cargar y dividir los datos de zoo dataset
    dataset = pd.read_csv('zoo.csv')
    X = dataset.drop(['animal_name', 'type'], axis=1)
    y = dataset['type']
    # Dividir los datos en conjuntos de entrenamiento y prueba
    X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.7)
    return X_train, X_test, y_train, y_test
```

Esta función carga los datos del archivo CSV y divide el conjunto en datos de entrenamiento y prueba.

Luego, se definen funciones individuales para cada modelo de clasificación, como logistic_regression(), k_nearest_neighbors(), support_vector_machine() y naive_bayes(). Estas funciones entrenan el modelo respectivo con los datos de entrenamiento y luego evalúan su desempeño utilizando la función evaluate_model(), que calcula diversas métricas como la precisión, la sensibilidad, la especificidad y la puntuación F1, además de graficar la matriz de confusión.

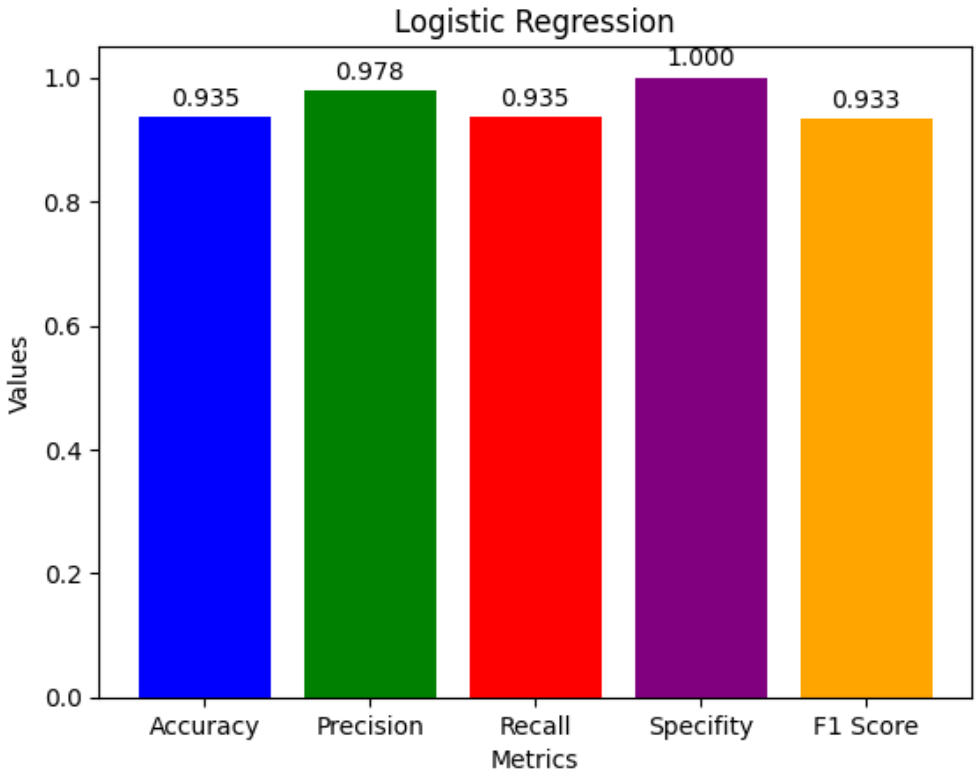
```
def evaluate_model(model_name, model, X_test, y_test):
    # Predicción en el conjunto de prueba
    y_pred = model.predict(X_test)

    # Cálculo de métricas
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted',
zero_division=1)
```

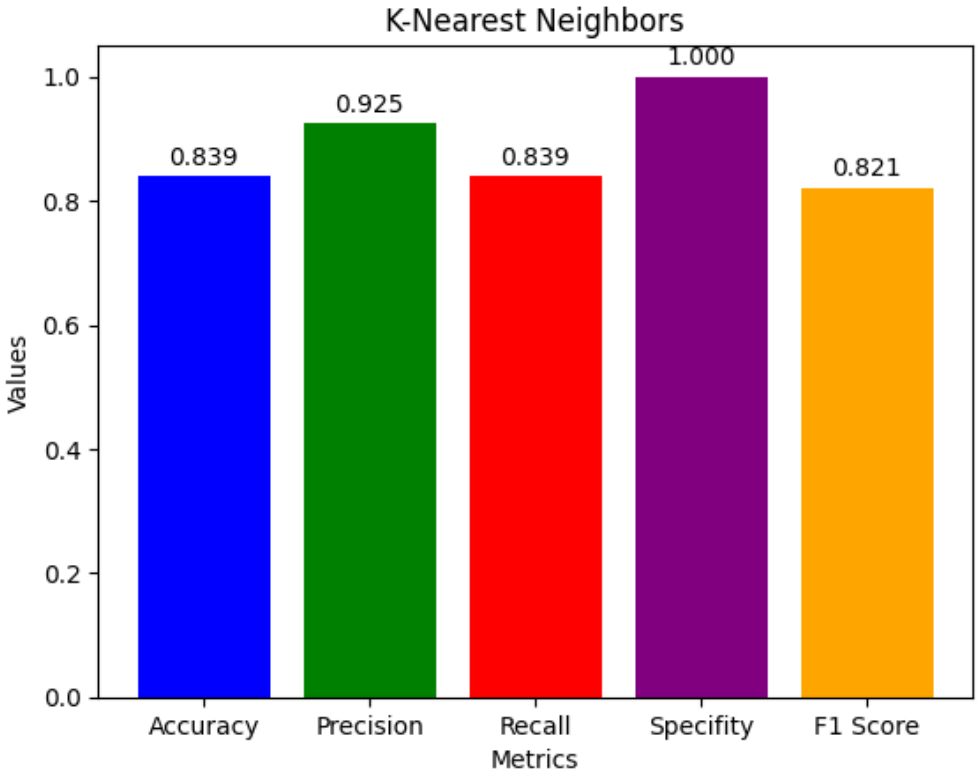
```
    recall = recall_score(y_test, y_pred, average='weighted',
zero_division=1)
    f1 = f1_score(y_test, y_pred, average='weighted')
    conf_matrix = confusion_matrix(y_test, y_pred)
    tn, fp, fn, tp = conf_matrix.ravel()[:4]
    specificity = tn / (tn + fp)

    # Imprimir métricas
    print(f"\n-------- {model_name} --------\n")
    print(f"Accuracy: {accuracy:.3f}")
    print(f"Precision: {precision:.3f}")
    print(f"Recall: {recall:.3f}")
    print(f"Specificity: {specificity:.3f}")
    print(f"F1 Score: {f1:.3f}\n")

    # Graficar las métricas
    metrics_names = ['Accuracy', 'Precision', 'Recall', 'Specifity',
'F1 Score']
    metrics_values = [accuracy, precision, recall, specificity, f1]
    plt.bar(metrics_names, metrics_values, color=['blue', 'green',
'red', 'purple', 'orange'])
    plt.title(model_name)
    plt.xlabel('Metrics')
    plt.ylabel('Values')
    # Añadir valores en cada barra
    for i, value in enumerate(metrics_values):
        plt.text(i, value + 0.01, f'{value:.3f}', ha='center',
va='bottom')
    plt.show()

def logistic_regression(X_train, X_test, y_train, y_test):
    model = LogisticRegression(max_iter=10000)
    model.fit(X_train, y_train)
    evaluate_model('Logistic Regression', model, X_test, y_test)

def k_nearest_neighbors(X_train, X_test, y_train, y_test):
    model = KNeighborsClassifier(n_neighbors=5)
    model.fit(X_train, y_train)
    evaluate_model('K-Nearest Neighbors', model, X_test, y_test)

def support_vector_machine(X_train, X_test, y_train, y_test):
    model = SVC(C=1.0)
    model.fit(X_train, y_train)
```

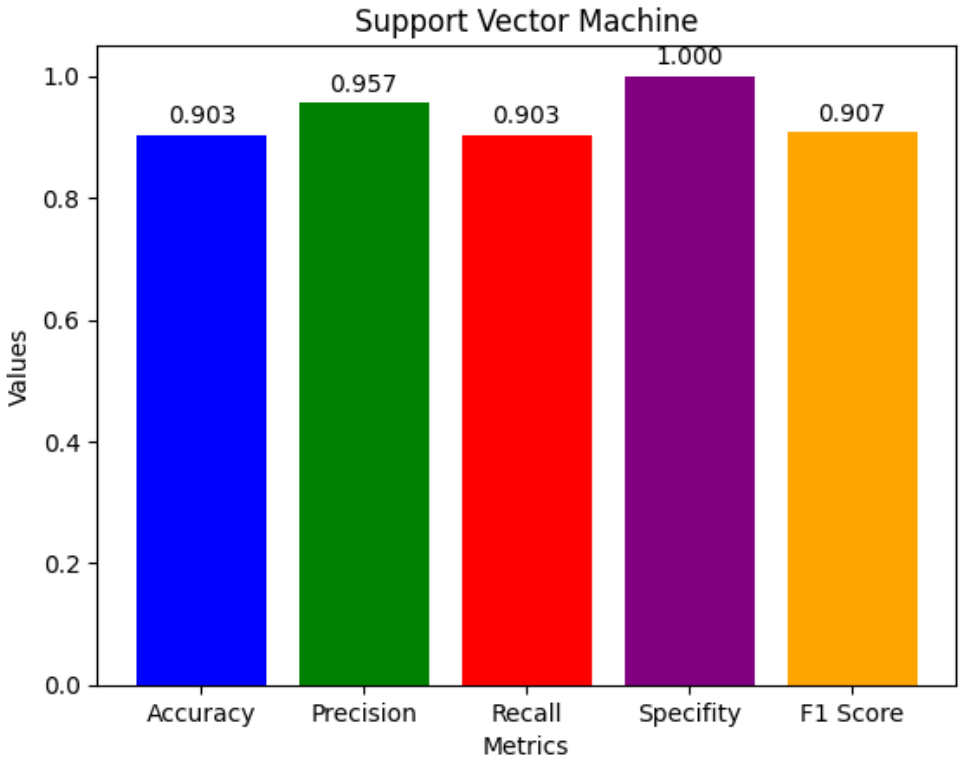
```
    evaluate_model('Support Vector Machine', model, X_test, y_test)

def naive_bayes(X_train, X_test, y_train, y_test):
    model = GaussianNB()
    model.fit(X_train, y_train)
    evaluate_model('Naive Bayes', model, X_test, y_test)
```

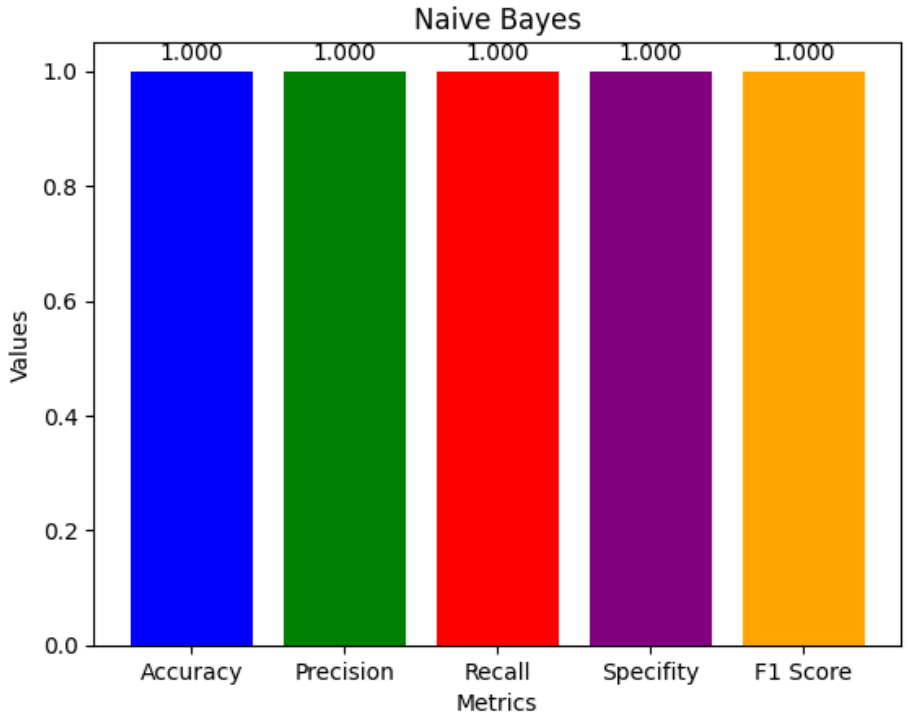
Resultados:

Modelo	Accuracy	Precision	Recall	Specificity	F1 Score
Regresión Logística	0.935	0.978	0.935	1.000	0.933
K-Vecinos Más Cercanos (KNN)	0.839	0.925	0.839	1.000	0.821
Máquinas de Vectores de Soporte (SVM)	0.903	0.957	0.903	1.000	0.907
Naive Bayes	1.000	1.000	1.000	1.000	1.000


Logistic Regression
0.935
0.978
0.935
1.000
0.933
1.0
0.8
0.6
0.4
0.2
0.0
Values
Accuracy
Precision
Recall
Specifity
F1 Score
Metrics


K-Nearest Neighbors
0.839
0.925
0.839
1.000
0.821
1.0
0.8
0.6
0.4
0.2
0.0
Values
Accuracy
Precision
Recall
Specifity
F1 Score
Metrics


Support Vector Machine
1.000
0.957
0.903
0.903
0.907
1.0
0.8
0.6
0.4
0.2
0.0
Values
Accuracy
Precision
Recall
Specify
F1 Score
Metrics


Naive Bayes
1.000
1.000
1.000
1.000
1.000
1.0
0.8
0.6
0.4
0.2
0.0
Values
Accuracy
Precision
Recall
Specify
F1 Score
Metrics

Conclusión:

Los resultados muestran que todos los clasificadores evaluados tienen un desempeño decente en la clasificación del conjunto de datos. La Regresión Logística, K-Vecinos Más Cercanos (KNN) y Máquinas de Vectores de Soporte (SVM) muestran una precisión y F1 score satisfactorias, aunque ligeramente inferiores a Naive Bayes, que alcanza una precisión perfecta en todas las métricas. Lo que indica que Naive Bayes es el modelo más efectivo para este conjunto de datos en particular. Sin embargo, la elección del modelo dependerá de las necesidades del problema y los requisitos de precisión.