Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
Seminario de Solución de Problemas de Inteligencia Artificial II
Dr. Diego Oliva
Depto. De Ciencias Computacionales
Proyecto final

Mercado Manzo Luis Alfonso

212559704

1. Objetivos

• Conocer otros clasificadores comúnmente usados en aprendizaje maquina

• Analizar diferentes datasets

• Conocer las diferencias entre los métodos de aprendizaje automático

• Identificar las ventajas y desventajas de cada método de clasificación

• Conocer e implementar las métricas para evaluar a los clasificadores

2. Actividades

Implementar los siguientes métodos y una red neuronal para clasificar el dataset de Zoo.

• Regresión logística (Logistic Regression)

• K-Vecinos Cercanos (K-Nearest Neighbors)

• Maquinas Vector Soporte (Support Vector Machines)

• Naive Bayes

Debes considerar que cada dataset tiene sus propias características por lo tanto tienes que generar un

modelo de aprendizaje automático con cada método. También debes tener en cuenta que en algunos

casos los archivos CSV deben ser generados por ti.

Evaluación de los resultados

Evaluar los resultados usando las siguientes métricas:

• Accuracy

• Precision

• Sensitivity

• Specificity

• F1 Score

Debes reportar los resultados en un documento PDF donde hagas una comparativa que te permita

decidir que método es el mejor para cada dataset. No olvides incluir tus conclusiones.

Codigo:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# Cargar el dataset
data = pd.read_csv('zoo.csv')

# Separar los datos en características (X) y la variable objetivo (y)
X = data.drop(['animal_name', 'class_type'], axis=1)  # Características
y = data['class_type']  # Variable objetivo

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)

# Inicializar los clasificadores
logistic_reg = LogisticRegression(max_iter=1000)
knn = KNeighborsClassifier(n_neighbors=5)
svm = SVC(kernel='linear')
naive_bayes = GaussianNB()

# Entrenar los modelos
logistic_reg.fit(X_train, y_train)
knn.fit(X_train, y_train)
svm.fit(X_train, y_train)
```

```
naive_bayes.fit(X_train, y_train)

# Realizar predicciones
y_pred_logistic = logistic_reg.predict(X_test)
y_pred_knn = knn.predict(X_test)
y_pred_svm = svm.predict(X_test)
y_pred_nb = naive_bayes.predict(X_test)

# Calcular las métricas para Logistic Regression
conf_matrix_logistic = confusion_matrix(y_test, y_pred_logistic)
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
precision_logistic = precision_score(y_test, y_pred_logistic, average='weighted')
recall_logistic = recall_score(y_test, y_pred_logistic, average='weighted')
f1_logistic = f1_score(y_test, y_pred_logistic, average='weighted')

# Calcular las métricas para K-Nearest Neighbors
conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
precision_knn = precision_score(y_test, y_pred_knn, average='weighted')
recall_knn = recall_score(y_test, y_pred_knn, average='weighted')
f1_knn = f1_score(y_test, y_pred_knn, average='weighted')

# Calcular las métricas para Support Vector Machines
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
precision_svm = precision_score(y_test, y_pred_svm, average='weighted')
recall_svm = recall_score(y_test, y_pred_svm, average='weighted')
f1_svm = f1_score(y_test, y_pred_svm, average='weighted')
```

```
# Calcular las métricas para Naive Bayes
conf_matrix_nb = confusion_matrix(y_test, y_pred_nb)
accuracy_nb = accuracy_score(y_test, y_pred_nb)
precision_nb = precision_score(y_test, y_pred_nb, average='weighted')
recall_nb = recall_score(y_test, y_pred_nb, average='weighted')
f1_nb = f1_score(y_test, y_pred_nb, average='weighted')

# Mostrar las métricas para cada modelo
print("Logistic Regression:")
print("Confusion Matrix:")
print(conf_matrix_logistic)
print(f"Accuracy: {accuracy_logistic}")
print(f"Precision: {precision_logistic}")
print(f"Sensitivity (Recall): {recall_logistic}")
print(f"F1 Score: {f1_logistic}")
print()

print("K-Nearest Neighbors:")
print("Confusion Matrix:")
print(conf_matrix_knn)
print(f"Accuracy: {accuracy_knn}")
print(f"Precision: {precision_knn}")
print(f"Sensitivity (Recall): {recall_knn}")
print(f"F1 Score: {f1_knn}")
print()

print("Support Vector Machines:")
print("Confusion Matrix:")
print(conf_matrix_svm)
```

```
print(f"Accuracy: {accuracy_svm}")
print(f"Precision: {precision_svm}")
print(f"Sensitivity (Recall): {recall_svm}")
print(f"F1 Score: {f1_svm}")
print()

print("Naive Bayes:")
print("Confusion Matrix:")
print(conf_matrix_nb)
print(f"Accuracy: {accuracy_nb}")
print(f"Precision: {precision_nb}")
print(f"Sensitivity (Recall): {recall_nb}")
print(f"F1 Score: {f1_nb}")
print()
```

resultados:

```
[[24  0  0  0  0  0  0]
 [ 0  6  0  0  0  0  0]
 [ 0  0  1  2  0  0  0]
 [ 0  0  0  5  0  0  0]
 [ 0  0  0  0  0  0  3]
 [ 0  0  0  0  0  5  0]
 [ 0  0  0  0  0  2  3]]
Accuracy: 0.8627450980392157
Precision: 0.8361344537815125
Sensitivity (Recall): 0.8627450980392157
F1 Score: 0.8345216874628638
```

K-Nearest Neighbors:

Confusion Matrix:

[[22 0 0 2 0 0 0]

[0 6 0 0 0 0 0]

[1 0 0 2 0 0 0]

[0 0 0 5 0 0 0]

[0 0 0 0 0 0 3]

[0 0 0 0 0 5 0]

[0 0 0 2 0 2 1]]

Accuracy: 0.7647058823529411

Precision: 0.706876031044829

Sensitivity (Recall): 0.7647058823529411

F1 Score: 0.722958095767858

Support Vector Machines:

Confusion Matrix:

[[24 0 0 0 0 0 0]

[0 6 0 0 0 0 0]

[0 0 1 1 1 0 0]

[0 0 0 5 0 0 0]

[0 0 0 0 3 0 0]

[0 0 0 0 0 5 0]

[0 0 0 0 0 1 4]]

Accuracy: 0.9411764705882353

Precision: 0.9526143790849673

Sensitivity (Recall): 0.9411764705882353

F1 Score: 0.933466315819257

Naive Bayes:

Confusion Matrix:

[[24 0 0 0 0 0 0]

[0 6 0 0 0 0 0]

[0 0 2 1 0 0 0]

[0 0 0 5 0 0 0]

[0 0 0 0 3 0 0]

[0 0 0 0 0 5 0]

[0 0 0 0 0 0 5]]

Accuracy: 0.9803921568627451

Precision: 0.9836601307189542

Sensitivity (Recall): 0.9803921568627451

F1 Score: 0.9793226381461676

Conclusion:

Para terminar, lo analizado entre esta actividad y la anterior, para mi opinion del metodo de evaluacion que me parece mas util es la de precision, me parece el mas adecuado al menos a las actividades realizadas. Aunque los otros metodos podrian ser utiles dependiendo de los datos que tengamos a nuestra disposicion.