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
Practica 2
Aprendizaje Maquina y Redes Neuronales

Ejercicio 2 y 3

Mercado Manzo Luis Alfonso

Código: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

Ejercicio 1. Realizar una investigación sobre los siguientes métodos de clasificación en aprendizaje

maquina:

• Regresión logística (Logistic Regression)

• K-Vecinos Cercanos (K-Nearest Neighbors)

• Maquinas Vector Soporte (Support Vector Machines)

• Naive Bayes

Debes identificar el funcionamiento de cada método y para que tipo de dataset es comúnmente usado.

Entregar la investigación en un archivo PDF.

Ejercicio 2. Implementar los métodos del Ejercicio 1 y alguna red neuronal para clasificar los

siguientes datasets:

• **Swedish Auto Insurance Dataset**

• **Wine Quality Dataset**

• **Pima Indians Diabetes Dataset**

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.

Ejercicio 3. Evaluar los resultados del Ejercicio 2 usando las siguientes métricas:

• Accuracy

• Precision

• Sensitivity

• Specificity

• F1 Score

Debes reportar los resultados de los Ejercicios 2 y 3 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.

Ejercicio 2:

Codigo:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
import sklearn.metrics as score
import pandas as pd
import numpy as np

# Funcion para calcular la especificidad
def specificity_score(y_test, preds):
    conf_matrix = score.confusion_matrix(y_test, preds)
    true_negatives = conf_matrix[0, 0]
    false_positives = conf_matrix[0, 1]
    return true_negatives / (true_negatives + false_positives)

def load_dataset(dataset_name):
    if dataset_name == "wine":
        dir_csv = 'vinos.csv'
        data = pd.read_csv(dir_csv, dtype=float)
        return data
    elif dataset_name == "diabetes":
        dir_csv = 'diabetes.csv'
        data = pd.read_csv(dir_csv, dtype=float, header=None)
        return data

    elif dataset_name == "seguro":
        dir_csv = 'autos.csv'
        data = pd.read_csv(dir_csv, dtype=float)
        return data
    else:
        return None

def perform_classification(data):
    if data is None:
        print("Dataset no válido.")
        return

    # Dividir los datos en X y Y
    X = np.array(data.iloc[:, :-1])  # Tomamos la última columna como
nuestro target
```

```
    y = np.array(data.iloc[:, -1])

    # Dividir en sets de prueba y entrenamiento
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)

    # Estandarizar los parámetros para knn y svm
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Entrenar clasificadores
    logistic_regression = LogisticRegression(max_iter=1000)
    knn = KNeighborsClassifier(n_neighbors=10)
    svm = SVC(kernel='linear')
    naive_bayes = GaussianNB()

    logistic_regression.fit(X_train, y_train)
    knn.fit(X_train, y_train)
    svm.fit(X_train, y_train)
    naive_bayes.fit(X_train, y_train)

    # Predicciones
    logistic_regression_pred = logistic_regression.predict(X_test)
    knn_pred = knn.predict(X_test)
    svm_pred = svm.predict(X_test)
    naive_bayes_pred = naive_bayes.predict(X_test)

    print("Comprobacion: ")
    for i in range(min(len(X_test), 10)):
        print(f"Valor real: {y_test[i]}")
        print(f"\tPredicción Regresión Logística:
{logistic_regression_pred[i]}")
        print(f"\tPredicción K Neighbors: {knn_pred[i]}")
        print(f"\tPredicción SVM: {svm_pred[i]}")
        print(f"\tPredicción Naive Bayes: {naive_bayes_pred[i]}")

def main():
    while True:
        print("\nSelecciona el dataset:")
        print("1. Calidad de Vinos")
        print("2. Diabetes")
        print("5. Seguros")
        print("6. Salir")
```

```
        choice = input("Ingresa el número correspondiente al dataset o
'3' para salir: ")

        if choice == '1':
            print("\n>>>>>vinos<<<<<")
            data = load_dataset("wine")
            perform_classification(data)
        elif choice == '2':
            print("\n>>>>>diabetes<<<<<<")
            data = load_dataset("diabetes")
            perform_classification(data)
        elif choice == '5':
            print("\n>>>>>>seguros<<<<<<<<")
            data = load_dataset("seguro")
            print("no se puede clasificar")
        elif choice == '6':
            print("Saliendo...")
            break
        else:
            print("Opción inválida. Por favor, elige una opción
válida.")

if __name__ == "__main__":
    main()
```

resultados:

vinos:

```
Selecciona el dataset:
1. Calidad de Vinos
2. Diabetes
5. Seguros
6. Salir
Ingresa el número correspondiente al dataset o '3' para salir: 1

>>>>>vinos<<<<<
Comprobacion:
Valor real: 5.0
      Predicción Regresión Logística: 5.0
      Predicción K Neighbors: 5.0
      Predicción SVM: 6.0
      Predicción Naive Bayes: 5.0
Valor real: 6.0
      Predicción Regresión Logística: 5.0
      Predicción K Neighbors: 6.0
      Predicción SVM: 6.0
      Predicción Naive Bayes: 6.0
```

```
Valor real: 6.0
      Predicción Regresión Logística: 7.0
      Predicción K Neighbors: 6.0
      Predicción SVM: 6.0
      Predicción Naive Bayes: 7.0
Valor real: 7.0
      Predicción Regresión Logística: 6.0
      Predicción K Neighbors: 7.0
      Predicción SVM: 6.0
      Predicción Naive Bayes: 7.0
Valor real: 7.0
      Predicción Regresión Logística: 6.0
      Predicción K Neighbors: 6.0
      Predicción SVM: 6.0
      Predicción Naive Bayes: 6.0
Valor real: 6.0
      Predicción Regresión Logística: 6.0
      Predicción K Neighbors: 6.0
      Predicción SVM: 6.0
      Predicción Naive Bayes: 6.0
Valor real: 5.0
      Predicción Regresión Logística: 6.0
      Predicción K Neighbors: 6.0
      Predicción SVM: 6.0
      Predicción Naive Bayes: 6.0
Valor real: 6.0
      Predicción Regresión Logística: 6.0
      Predicción K Neighbors: 6.0
      Predicción SVM: 6.0
      Predicción Naive Bayes: 6.0
Valor real: 6.0
      Predicción Regresión Logística: 5.0
      Predicción K Neighbors: 6.0
      Predicción SVM: 5.0
      Predicción Naive Bayes: 5.0
Valor real: 6.0
      Predicción Regresión Logística: 6.0
      Predicción K Neighbors: 7.0
      Predicción SVM: 6.0
      Predicción Naive Bayes: 8.0
```

Diabetes:

```
>>>>>diabetes<<<<<<
Comprobacion:
Valor real: 0.0
      Predicción Regresión Logística: 0.0
      Predicción K Neighbors: 0.0
      Predicción SVM: 0.0
      Predicción Naive Bayes: 0.0
Valor real: 0.0
      Predicción Regresión Logística: 0.0
      Predicción K Neighbors: 0.0
```

```
      Predicción SVM: 0.0
      Predicción Naive Bayes: 0.0
Valor real: 0.0
      Predicción Regresión Logística: 0.0
      Predicción K Neighbors: 0.0
      Predicción SVM: 0.0
      Predicción Naive Bayes: 1.0
Valor real: 1.0
      Predicción Regresión Logística: 1.0
      Predicción K Neighbors: 0.0
      Predicción SVM: 1.0
      Predicción Naive Bayes: 1.0
Valor real: 0.0
      Predicción Regresión Logística: 0.0
      Predicción K Neighbors: 0.0
      Predicción SVM: 0.0
      Predicción Naive Bayes: 0.0
Valor real: 1.0
      Predicción Regresión Logística: 1.0
      Predicción K Neighbors: 0.0
      Predicción SVM: 1.0
      Predicción Naive Bayes: 1.0
Valor real: 1.0
      Predicción Regresión Logística: 1.0
      Predicción K Neighbors: 1.0
      Predicción SVM: 1.0
      Predicción Naive Bayes: 1.0
Valor real: 0.0
      Predicción Regresión Logística: 0.0
      Predicción K Neighbors: 0.0
      Predicción SVM: 0.0
      Predicción Naive Bayes: 0.0
Valor real: 0.0
      Predicción Regresión Logística: 0.0
      Predicción K Neighbors: 0.0
      Predicción SVM: 0.0
      Predicción Naive Bayes: 0.0
Valor real: 1.0
      Predicción Regresión Logística: 1.0
      Predicción K Neighbors: 1.0
      Predicción SVM: 1.0
      Predicción Naive Bayes: 0.0

Seguros:

>>>>>>seguros<<<<<<<
no se puede clasificar
```
(por mas que intente hacerlo no pude conseguir resultados.)

Ejercicio 3:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
import sklearn.metrics as score
import pandas as pd
import numpy as np

# Funcion para calcular la especificidad
def specificity_score(y_test, preds):
    conf_matrix = score.confusion_matrix(y_test, preds)
    true_negatives = conf_matrix[0, 0]
    false_positives = conf_matrix[0, 1]
    return true_negatives / (true_negatives + false_positives)

def load_dataset(dataset_name):
    if dataset_name == "wine":
        dir_csv = 'vinos.csv'
        data = pd.read_csv(dir_csv, dtype=float)
        return data
    elif dataset_name == "diabetes":
        dir_csv = 'diabetes.csv'
        data = pd.read_csv(dir_csv, dtype=float, header=None)
        return data
    elif dataset_name == "seguro":
        dir_csv = 'autos.csv'
        data = pd.read_csv(dir_csv, dtype=float)
        return data
    else:
        return None

def perform_classification(data):
    if data is None:
        print("Dataset no válido.")
        return

    # Dividir los datos en X y Y
    X = np.array(data.iloc[:, :-1])  # Tomamos la última columna como
nuestro target
    y = np.array(data.iloc[:, -1])
```

```
    # Dividir en sets de prueba y entrenamiento
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)

    # Estandarizar los parámetros para knn y svm
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Entrenar clasificadores
    logistic_regression = LogisticRegression(max_iter=1000)
    knn = KNeighborsClassifier(n_neighbors=10)
    svm = SVC(kernel='linear')
    naive_bayes = GaussianNB()

    logistic_regression.fit(X_train, y_train)
    knn.fit(X_train, y_train)
    svm.fit(X_train, y_train)
    naive_bayes.fit(X_train, y_train)

    # Predicciones
    logistic_regression_pred = logistic_regression.predict(X_test)
    knn_pred = knn.predict(X_test)
    svm_pred = svm.predict(X_test)
    naive_bayes_pred = naive_bayes.predict(X_test)

    # Métricas para Regresión Logística
    print("Métricas para Regresión Logística:")
    print("Accuracy:", score.accuracy_score(y_test,
logistic_regression_pred))
    print("F1 Score:", score.f1_score(y_test,
logistic_regression_pred, average='weighted'))

    # Métricas para KNN
    print("\nMétricas para K Nearest Neighbors:")
    print("Accuracy:", score.accuracy_score(y_test, knn_pred))
    print("F1 Score:", score.f1_score(y_test, knn_pred,
average='weighted'))

    # Métricas para SVM
    print("\nMétricas para Support Vector Machine:")
    print("Accuracy:", score.accuracy_score(y_test, svm_pred))
    print("F1 Score:", score.f1_score(y_test, svm_pred,
average='weighted'))
```

```
    # Métricas para Naive Bayes
    print("\nMétricas para Naive Bayes:")
    print("Accuracy:", score.accuracy_score(y_test, naive_bayes_pred))
    print("F1 Score:", score.f1_score(y_test, naive_bayes_pred,
average='weighted'))

def main():
    while True:
        print("\nSelecciona el dataset:")
        print("1. Calidad de Vinos")
        print("2. Diabetes")
        print("3. Seguros")
        print("4. Salir")

        choice = input("Ingresa el número correspondiente al dataset o
'4' para salir: ")


        if choice == '1':
            print("\n>>>>>vinos<<<<<")
            data = load_dataset("wine")
            perform_classification(data)
        elif choice == '2':
            print("\n>>>>>diabetes<<<<<<")
            data = load_dataset("diabetes")
            perform_classification(data)
        elif choice == '5':
            print("\n>>>>>>seguros<<<<<<<<")
            data = load_dataset("seguro")
            print("no se puede clasificar")
        elif choice == '6':
            print("Saliendo...")
            break
        else:
            print("Opción inválida. Por favor, elige una opción
válida.")

if __name__ == "__main__":
    main()
```

resultados:

```
Selecciona el dataset:
1. Calidad de Vinos
```

```
2. Diabetes
3. Seguros
4. Salir
Ingresa el número correspondiente al dataset o '4' para salir: 1

>>>>>vinos<<<<<
Métricas para Regresión Logística:
Accuracy: 0.523469387755102
F1 Score: 0.483969391969923

Métricas para K Nearest Neighbors:
Accuracy: 0.536734693877551
F1 Score: 0.5143672112314547

Métricas para Support Vector Machine:
Accuracy: 0.5091836734693878
F1 Score: 0.4269009928630747

Métricas para Naive Bayes:
Accuracy: 0.4561224489795918
F1 Score: 0.443212510968493

Selecciona el dataset:
1. Calidad de Vinos
2. Diabetes
3. Seguros
4. Salir
Ingresa el número correspondiente al dataset o '4' para salir: 2

>>>>>diabetes<<<<<<
Métricas para Regresión Logística:
Accuracy: 0.7207792207792207
F1 Score: 0.6971692996896851

Métricas para K Nearest Neighbors:
Accuracy: 0.6818181818181818
F1 Score: 0.6606737228746797

Métricas para Support Vector Machine:
Accuracy: 0.7272727272727273
F1 Score: 0.7102797202797204

Métricas para Naive Bayes:
Accuracy: 0.7337662337662337
F1 Score: 0.7254813325533901

Selecciona el dataset:
1. Calidad de Vinos
2. Diabetes
3. Seguros
4. Salir
Ingresa el número correspondiente al dataset o '4' para salir: 5

>>>>>>seguros<<<<<<<<
```

```
no se puede clasificar

Selecciona el dataset:
1. Calidad de Vinos
2. Diabetes
3. Seguros
4. Salir
```