

HDT 7 RNA y SVM

RNA

1. Genere dos modelos de redes neuronales que sea capaz de clasificar usando la variable respuesta que categoriza las casas en baratas, medias y caras. Estos modelos deben tener diferentes topologías y funciones de activación.
2. Use los modelos para predecir el valor de la variable respuesta
3. Haga las matrices de confusión respectivas.

Matriz de confusión para la variable Clasificación
Red Neuronal #1

```
Accuracy: 1.0  
Precision: 1.0  
Recall: 1.0  
Matriz de confusión: [[390  0  0]  
 [ 0 47  0]  
 [ 0  0 1]]
```

✓ 0.2s

Matriz de confusión para la variable Clasificación
Red Neuronal #2 (activación logística)

```
Accuracy: 0.997716894977169  
Precision: 0.997716894977169  
Recall: 0.997716894977169  
Matriz de confusión: [[390  0  0]  
 [ 0 47  0]  
 [ 0  1  0]]
```

✓ 0.9s

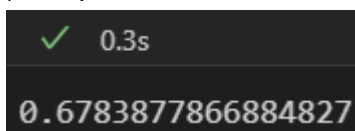
4. Compare los resultados obtenidos con los diferentes modelos de clasificación usando redes neuronales en cuanto a efectividad, tiempo de procesamiento y equivocaciones (donde el algoritmo se equivocó más, donde se equivocó menos y la importancia que tienen los errores).

En cuanto a tiempos de procesamiento podemos ver que la red neuronal #1 toma mucho menor tiempo y es más efectiva, ya que en la segunda red neuronal podemos ver un "1" en una columna que no corresponde, lo que significa que existe error en la clasificación. A pesar de que solamente es un dato el que presentó errores, el algoritmo #2 muestra fallas.

5. Seleccione ahora el SalesPrice como variable respuesta.
6. Genere dos modelos de regresión con redes neuronales con diferentes topologías y funciones de activación para predecir el precio de las casas.

Red Neuronal #1

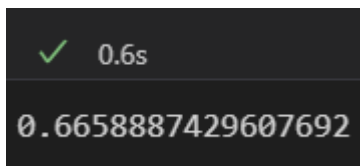
(Sin aplicar funciones de activación)



✓ 0.3s
0.6783877866884827

Red Neuronal #2

Red Neuronal #2 (activación logística)



✓ 0.6s
0.6658887429607692

7. Compare los dos modelos de regresión y determine cuál funciona mejor para predecir el precio de las casas.

El modelo #1, el cual analiza la variable Clasificación, funciona mejor para predecir el precio de las casas, debido a que esta es una variable categórica. Por lo tanto, el modelo #2, que analiza directamente el precio de las casas y que es una variable cuantitativa, no funciona muy bien para este análisis. Esto se puede ver en los scores obtenidos para cada modelo.

8. Compare la eficiencia del mejor modelo de RNA con los resultados obtenidos con los algoritmos de las hojas de trabajo anteriores. ¿Cuál es mejor para predecir? ¿Cuál se demoró más en procesar?

Como podemos ver en cuanto a la eficiencia, no aplicar funciones de activación reduce el tiempo de ejecución. Si comparamos estos scores con scores de hojas anteriores, podemos ver que este algoritmo es el que peor score presenta si se trata de analizar la variable de respuesta de sales price de forma directa, por otro lado al analizar una variable categórica relacionada a la variable de respuesta, este modelo es el mejor modelo hasta el momento.

9. Compare los resultados del mejor modelo de esta hoja para clasificar con los resultados de los algoritmos usados para clasificar de las hojas de trabajo anteriores

Este modelo no se adaptó a los datos de la mejor forma y se puede notar con el score que presentó siendo 0.66-0.67, cuando los modelos de las otras hojas en promedio dieron arriba de 0.8 lo cual nos indica que este modelo no es el acertado para analizar y predecir cuando se trata de analizar la variable de respuesta de forma directa, pero si se trata de analizar la variable categórica de clasificación que va relacionada con la clasificación este modelo es el mejor modelo disponible a comparación de las hojas anteriores, con un score muy cercano a 1 siendo 0.99, a pesar de que tiene overfitting, debido que no se limpiaron o normalizaron los datos de forma correcta, este score sigue siendo el mejor comparado con los demás.

10. Compare los resultados del mejor modelo para predecir el precio de venta con los resultados de los algoritmos usados para el mismo propósito de las hojas de trabajo anteriores.

Este modelo no se adaptó a los datos de la mejor forma y se puede notar con el score que presentó. Existen mejores modelos para representar este conjunto de datos en específico. Como se mencionó anteriormente este modelo de RNA es excelente al analizar una variable categórica de modo de clasificación, a pesar de que el score del mejor modelo es de 1 lo cual indica un overfitting elevado se denota que con la normalización de los datos correcta este tendrá un score elevado, y es el mejor modelo por diferencia comparado con los demás, por otro lado si se analiza de forma directa la variable de respuesta este será el peor modelo.

11. Genere un informe de los resultados y las explicaciones.

Las preguntas anteriores lo resumen.

SVM

1. Explore los datos y explique las transformaciones que debe hacer para generar un modelo de máquinas vectoriales de soporte.

Se hizo el análisis exploratorio como en las hojas de trabajo anteriores, se eliminaron las variables que no aportan valor en la matriz de correlación. Por otro lado se convirtieron variables de tipo texto a número, además se agregaron valores faltantes.

2. Genere varios (más de 2) modelos de SVM con diferentes kernels y distintos valores en los parámetros c, gamma y d (en caso de que utilice el polinomial).

Kernel: Poly, C=0.5, Gamma: Scale

```
clf_svm = SVC(kernel='poly', C=0.5, gamma='scale')
modelo_2 = clf_svm.fit(X_train_scaled, y_train)
plot_confusion_matrix(clf_svm,
                       X_test_scaled,
                       y_test,
                       display_labels=["Barata", "Intermedia", "Cara"])
```

Kernel: rbf, C=10, Gamma: Auto

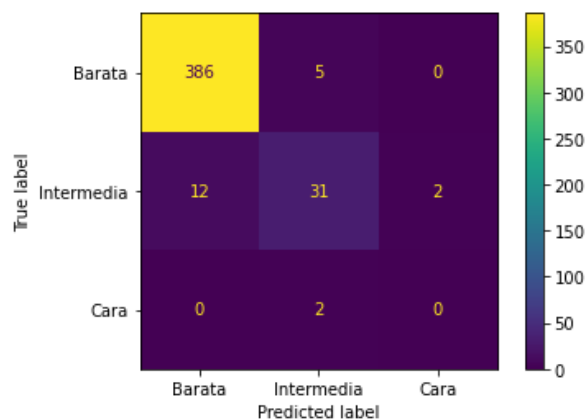
```
clf_svm = SVC(kernel='rbf', C=10, gamma='auto')
modelo_3 = clf_svm.fit(X_train_scaled, y_train)

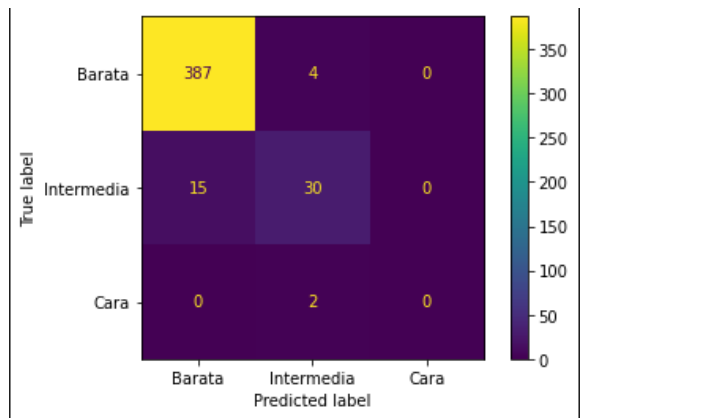
plot_confusion_matrix(
    clf_svm,
    X_test_scaled,
    y_test,
    display_labels=["Barata", "Intermedia", "Cara"]
)
```

Kernel: linear, C=1, Gamma: 1

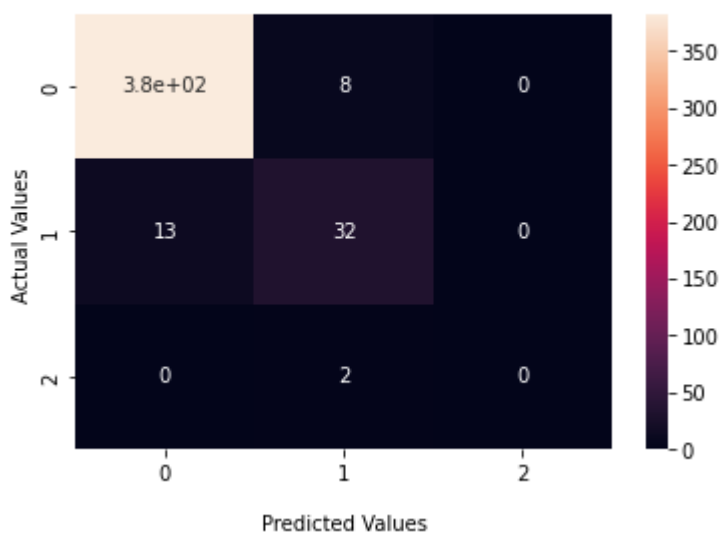
```
clf_svm = SVC(kernel='linear', C=1, gamma=1)
modelo_1 = clf_svm.fit(X_train_scaled, y_train)
plot_confusion_matrix(clf_svm,
                       X_test_scaled,
                       y_test,
                       display_labels=["Barata", "Intermedia", "Cara"])
```

3. Haga las matrices de confusión respectivas.





Matriz de confusión



4. Compare los resultados obtenidos con los diferentes modelos que hizo en cuanto a efectividad, tiempo de procesamiento y equivocaciones (donde el algoritmo se equivocó más, donde se equivocó menos y la importancia que tienen los errores).

El modelo con Kernel: linear, C=1, Gamma: 1

Duro 0.3s, con una efectividad de 0.952, se equivocó en un total de 21 casas para clasificar

El modelo con Kernel: Poly, C=0.5, Gamma: Scale

Duro 0.3s, con una efectividad de 0.942, se equivocó en un total de 25 casas para clasificar

El modelo con Kernel: rbf, C=10, Gamma: Auto

Duro 0.2s, con una efectividad de 0.952, se equivocó en un total de 19 casas para clasificar

El modelo con Kernel: rbf, C=100, Gamma: 0.01

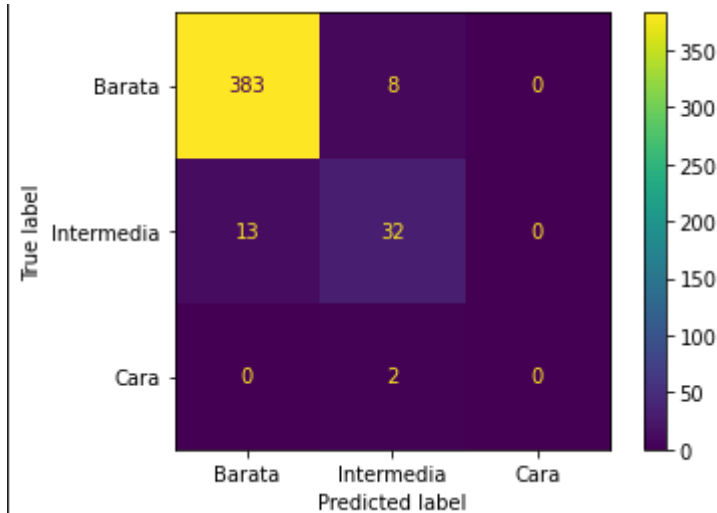
Duro 0.4s, con una efectividad de 0.947, se equivocó en un total de 23 casas para clasificar

Este último modelo tiene los parámetros obtenidos luego de ejecutar el GridSearchCV que nos indica cuales son los mejores parámetros para el modelo, sin embargo no dio los mejores datos en cuanto efectividad y clasificación. Esto se debe a los diferentes tipos de kernel, por este va variando los resultados.

5. Compare la eficiencia del mejor modelo de SVM con los resultados obtenidos en los algoritmos de las hojas de trabajo anteriores que usen la misma variable respuesta (árbol de decisión y random forest, naive bayes). ¿Cuál es mejor para predecir? ¿Cuál se demoró más en procesar?

El mejor es el rbf por la clasificación que se hace antes de grid search, la cual dio los parámetros de $C = 100$, $\gamma = 0.01$ Kernel: rbf.

Esta es la mejor porque tuvo menos fallo al clasificar a comparación de los algoritmos de clasificación anteriores, además duró 0.2s en ejecutarse



6. Genere un informe de los resultados y las explicaciones.
Las preguntas anteriores lo resumen.