Aprendizaje Automático
Práctica 1

1) **Análisis exploratorio de los datos**

   Importa los datos del fichero `dataset.csv` y realiza el análisis exploratorio de los datos. Describe en el informe los resultados de este análisis y deposita el código Python en Aula Virtual en el fichero answer1.ipynb

2) **Clasificación usando todas las características de los datos**

   Usando todas las características, implementa los métodos

   - Logistic Regression,
   - SVM, y
   - Random Trees

   para clasificar los datos. Describe en en informe los parámetros usados y los resultados obtenidos con los distintos métodos y deposita el código Python en Aula Virtual en el fichero answer2.ipynb

3) **Clasificación usando 4 características de los datos**

   Selecciona 4 características de los datos. Usando estas características, implementa los métodos

   - Logistic Regression,
   - SVM, y
   - Random Trees

   para clasificar los datos. Describe en en informe los parámetros usados y los resultados obtenidos con los distintos métodos y deposita el código Python en Aula Virtual en el fichero answer3.ipynb

4) **Clasificación usando 2 características de los datos**

   Selecciona 2 características de los datos. Usando estas características, implementa los métodos

   - Logistic Regression,
   - SVM, y
   - Random Trees

   para clasificar los datos. Describe en en informe los parámetros usados y los resultados obtenidos con los distintos métodos y deposita el código Python en Aula Virtual en el fichero answer4.ipynb.

Redacta un informe detallado respondiendo a cada pregunta en una sección diferente. Motiva cada elección realizada durante el proceso de diseño de los clasificadores y describe cada resultado obtenido. Compara los resultados obtenidos con distintos números de características. Incluye el código Python en el apartado correspondiente del informe.

1 Análisis exploratorio de los datos

1.1 Importacion del dataset

El primer paso de este analisis consiste en la importacion del dataset, utilizando la libreria pandas para leer el fichero dataset.csv. A partir de ahi, se configura la visualizacion del dataset para poder observar todas las columnas disponibles.

Este paso es importante para obtener una vista inicial del tamaño y las caracteristicas del dataset, ademas de confirmar que los datos han sido leidos correctamente.

```
dataset = pd.read_csv("dataset.csv")
pd.set_option('display.max_columns', len(dataset.columns))
```

1.2 Inspeccion del dataset

A continuacion, se evalua la estructura del dataset observando las dimensiones (dataset.shape), las primeras filas (dataset.head(1), la informacion de cada columna y otras estadisticas descriptivas como la media, la desviacion estandar, minimos, maximos y cuartiles, ademas de otros valores atipicos y distribuciones inusuales (dataset.describe()).

Todo esto permite conocer cuantas filas y columnas contiene el dataset (dataset.columns), el tipo de datos que maneja (dataset.info()), la aparicion de valores nulos o la presencia de inconsistencia en algun dato.

```
dataset.columns
dataset.shape
dataset.head(1)
dataset.info()
dataset.describe()
```

1.3 Anonimizacion de los datos y analisis de correlacion

Para poder analizar los datos, se ha creado un nuevo conjunto de datos anonimizado, en el que se ha eliminado la columna Target del dataset. Ademas de esto, se realiza un analisis de la correlacion entre todas las columnas/variables.

Es importante realizar este paso para detectar altas correlaciones, las cuales suelen indicar multicolinealidad, algo que afectaria a los modelos predictivos.

```
dataset_anonymized = dataset.drop(["Target"], axis=1)
dataset_anonymized.to_csv('dataset_anonymized.csv', index=False)
dataset_anonymized.corr()
```

El analisis de una correlacion esta identificada con las demas, donde un coeficiente de correlacion cercano a 1 o -1 indica una relacion fuerte, mientras que un valor cercano a 0 indica una correlacion practicamente inexistente.

1.4 Visualizacion de la matriz de correlacion

Una matriz de correlacion es visualizada mediante un mapa de calor utilizando la libreria seaborn, la cual facilita la identificacion de relaciones positivas o negativas entre las variables.

```
fig, ax = plt.subplots(figsize=(9,9))
sb.heatmap(dataset.corr(), linewidth=0.5, annot=True)
```

El grafico resultante proporciona una vision mas clara de las relaciones mas fuertes entre las variables, donde los colores oscuros indican una correlacion alta, mientras que los colores mas claros indican una correlacion baja.

IMAGEN MAPA DE CALOR CORRELACIONES

1.5 **Visualizacion de distribuciones**

Para observar la distribucion de cada una de las variables en el dataset anonimizado, se generan histogramas que permiten identificar patrones y verificar si las variables estan distribuidas de manera normal, poseen sesgos o son valores atipicos.

```
columns = dataset_anonymized.columns
fig = plt.figure(figsize=(12,12))
for i in range(0,11):
  ax = plt.subplot(4,4,i+1)
  ax.hist(dataset_anonymized[columns[i]], bins=20, color='blue', edgecolor='black')
  ax.set_title(dataset_anonymized.head(0)[columns[i]].name)
plt.tight_layout()
plt.show()
```

Estos histogramas muestran la frecuencia de los valores dentro de ciertos intervalos, algo que ayuda a entender mejor la distribucion de los datos.

IMAGEN HISTOGRAMA

1.6 **Separacion de caracteristicas y etiquetas**

Una vez hecho esto, se separan las caracteristicas (X) y las etiqeutas (y) del dataset, lo cual permite preparar los datos para los modelos de machine learning.

En este caso, X contiene la informacion de todas las columnas, exceptuando la columna Target, mientras que y contiene los valores de la variable objetivo Target.

```
X = dataset_anonymized
y = dataset.get("Target")
print('Class labels:', np.unique(y))

# RESULTADO
Class labels: [3 4 5 6 7 8]
```

1.7 **Division del dataset en entrenamiento y prueba**

El dataset se divide en dos conjuntos: Uno de entrenamiento (0.75) y otro de prueba (0.25). Esta division se hace de tal forma que la proporcion de clases en ambos conjuntos sea similar, evitando asi posibles desbalanceos.

Es necesario realizar este paso para poder entrenar los modelos con una parte del dataset y luego probar su rendimiento con datos que no han sido vistos previamente.

```
X_train, X_test, y_train, y_test = train_test_split(X.values, y, test_size=0.25,
 random_state=1, stratify=y)
```

1.8 **Estandarizacion de los datos**

La estandarizacion de los datos asegura que todas las caracteristicas se encuentren en la misma escala, algo que es considerado de gran importancia en modelos de regresion logistica, SVM o redes neuronales, donde se veran algunos de ellos mas adelante.

Para este caso, se utiliza StandardScaler() para normalizar los datos, los cuales se ajustan a las caracteristicas de los mismos para que tengan media 0 y desviacion estandar 1, lo que se traduce en una mejora de rendimiento a la hora de utilizar los algoritmos mencionados anteriormente.

```
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

1.9 **Verificacion del balance de clases**

Y por ultimo, se evalua el balance de las clases tanto en los datos de entrenamiento como en los datos de prueba para verificar si existe algun tipo de desbalance en las etiquetas, ya que de darse este caso, se podria producir un desbalance significativo que puede afectar a la capacidad predictiva del modelo.

```
print('Labels counts in y:', np.bincount(y))
print('Labels counts in y_train:', np.bincount(y_train))
print('Labels counts in y_test:', np.bincount(y_test))

# RESULTADO
Labels counts in y:       [  0   0   0  10  53  681   638   199   18  ]
Labels counts in y_train: [  0   0   0   8  40  511   478   149   13  ]
Labels counts in y_test:  [  0   0   0   2  13  170   160    50    5  ]
```

Es importante mencionar que el balance de clases da una idea de si existe predominancia de una clase sobre otra, algo que podria requerir tecnicas de remuestreo para mitigar el sesgo.

2 Clasificación usando todas las características de los datos

2.1 Introduccion

En este apartado se presentan los resultados de la clasificacion del conjunto de datos del fichero dataset.csv utilizando 3 metodos de aprendizaje supervisado: Regresion Logistica (Logistic Regression), Maquinas de Soporte Vectorial (SVM) y Arboles de Decision (Random Trees).

Cada uno de estos clasificadores ha sido entrenado usando todas las caracteristicas del dataset. A continuacion, se iran describiendo los parametros utilizados para cada modelo, presentando los resultados obtenidos y discutiendo el rendimiento de cada clasificador.

2.2 Carga y preparacion de los datos

Antes de entrenar los modelos, primero se debe cargar el dataset, la anonimizacion de las caracteristicas y la separacion de las caracteristicas (X) respecto a la etiqueta objetivo (y).

Ademas de esto, tambien se realiza una estandarizacion de las caracteristicas, algo que es muy importante realizar para modelos como la Regresion Logistica (Logistic Regression) y SVM (Support Vector Machines), que son sensibles a las escalas de las variables.

```
dataset = pd.read_csv("dataset.csv")
dataset_all_characteristics = dataset.drop(["Target"], axis=1)
X = dataset_all_characteristics
y = dataset.get("Target")

X_train, X_test, y_train, y_test = train_test_split(X.values, y, test_size=0.25,
random_state=1, stratify=y)
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

Por otro lado, el dataset se divide en dos conjuntos: Uno de entrenamiento (0.75) y otro de prueba (0.25). Esta division se hace de tal forma que la proporcion de clases en ambos conjuntos sea similar, evitando asi posibles desbalanceos.

2.3 Regresion Logistica (Logistic Regression)

La Regresion Logistica es un modelo lineal, por lo que es mas adecuado cuando las caracteristicas tienen relaciones lineales con las etiquetas. En este caso, el modelo muestra un buen rendimiento general, aunque la precision podria mejorar utilizando tecnicas de seleccion de caracteristicas o ajustando mas los parametros.

A continuacion, se explican brevemente los parametros utilizados:

- C: El parametro de regularizacion C controla la penalizacion aplicada a los errores. Un valor alto de C indica que se permite una menor penalizacion, por lo que el modelo intenta ajustar mas los datos.
- solver: El solver 'lbfgs' es un optimizador recomendado para solucionar problemas pequeños y medianos.
- multi-class: Procedente de las siglas OnevsRest (OVR), significa que para la clasificacion multiclase, el modelo entrenara un clasificador idenpendiente para cada clase.

```
lr = LogisticRegression(C=100.0, solver='lbfgs', multi_class='ovr')
lr.fit(X_train_std, y_train)
y_pred = lr.predict(X_test_std)
print('Misclassification samples: %d' % (y_test != y_pred).sum())
print('Accuracy: %.3f' % lr.score(X_test_std, y_test))

# RESULTADO
Misclassification samples: 165
Accuracy. 0.588
```

Se ha obtenido como resultado 165 muestras mal clasificadas, un numero relativamente bajo para este modelo, y se obtuvo una precision de 0.588, por lo que esa proporcion de predicciones del modelo fur correcta.

2.4 **Maquinas de Soporte Vectorial (SVM)**

SVM con kernel RBF es potente en la captura de relaciones no lineales entre las características. El modelo muestra un rendimiento sólido y, aunque los resultados dependen mucho de la selección de los parámetros gamma y C, en este caso el ajuste da buenos resultados.

A continuacion, se explican brevemente los parametros utilizados:

- kernel = 'rbf': Se utiliza el kernel radial base (RBF), que es adecuado para problemas no lineales.
- gamma = 0.7: Controla el grado de influencia de los puntos individuales. Un valor bajo significa que el área de influencia de cada punto es alta, mientras que un valor alto restringe el área.
- C = 30.0: Controla el grado de penalización aplicado a los errores de clasificación. Un valor más alto de C tiende a reducir los errores de clasificación en el conjunto de entrenamiento.

```
svm = SVC(kernel='rbf', random_state=1, gamma=0.7, C=30.0)
svm.fit(X_train_std, y_train)
y_pred = svm.predict(X_test_std)
print('Misclassification samples: %d' % (y_test != y_pred).sum())
print('Accuracy: %.3f' % svm.score(X_test_std, y_test))

# RESULTADO
Misclassification samples: 157
Accuracy. 0.608
```

Se ha obtenido como resultado 157 muestras mal clasificadas, un numero relativamente bajo para este modelo, y se obtuvo una precision de 0.608, una proporcion ligeramente superior a la obtenida con la Regresion Logistica.

2.5 **Arboles de Decision (Random Trees)**

Los árboles de decisión tienden a sobreajustar los datos si no se limitan adecuadamente. En este caso, al limitar la profundidad a 4, se puede evitar el sobreajuste y el modelo puede ser generalizado adecuadamente. Sin embargo, en comparación con los otros modelos, el rendimiento fue ligeramente inferior, posiblemente porque el modelo no pudo capturar todas las complejidades de los datos con solo 4 niveles de profundidad.

A continuacion, se explican brevemente los parametros utilizados:

- criterion = 'gini': Utiliza el índice de Gini para medir la pureza de los nodos.
- max-depth = 4: La profundidad máxima del árbol se fija en 4 para evitar el sobreajuste.

```
tree_model = DecisionTreeClassifier(criterion='gini', max_depth=4, random_state=1)
tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)
print('Misclassification samples: %d' % (y_test != y_pred).sum())
print('Accuracy: %.3f' % tree_model.score(X_test, y_test))

# Visualización del árbol de decisión
tree.plot_tree(tree_model, feature_names=['Col1', 'Col2', 'Col3', 'Col4', 'Col5', 'Col6', 'Col7',
'Col8', 'Col9', 'Col10', 'Col11'], filled=True)
plt.show()

# RESULTADO
Misclassification samples: 172
Accuracy. 0.570
```

Se ha obtenido como resultado 172 muestras mal clasificadas, un numero similar a los otros dos modelos, y se obtuvo una precision de 0.570, por lo que esa proporcion de predicciones del modelo fur correcta.

3 Clasificación usando 4 características de los datos

3.1 Introduccion

En este apartado se presentan los resultados de la clasificacion del conjunto de datos del fichero dataset.csv utilizando 3 metodos de aprendizaje supervisado: Regresion Logistica (Logistic Regression), Maquinas de Soporte Vectorial (SVM) y Arboles de Decision (Random Trees).

Cada uno de estos clasificadores ha sido entrenado usando todas las caracteristicas del dataset. A continuacion, se iran describiendo los parametros utilizados para cada modelo, presentando los resultados obtenidos y discutiendo el rendimiento de cada clasificador.

3.2 Seleccion de las 4 caracteristicas

En primer lugar, se realizó un análisis de correlación entre las variables del dataset completo para identificar las características más relevantes. Se seleccionaron las siguientes cuatro características basadas en su diversidad y posible relación con la variable objetivo:

Col1 Col5 Col7 Col11

Estas variables se seleccionaron por su correlación baja entre sí y por su posible relevancia para la clasificación, a fin de evitar multicolinealidad y maximizar la capacidad predictiva del modelo.

```
CODE
```

3.3 Preparacion de los datos

Antes de entrenar los modelos, primero se debe cargar el dataset, la anonimizacion de las caracteristicas y la separacion de las caracteristicas (X) respecto a la etiqueta objetivo (y).

Ademas de esto, tambien se realiza una estandarizacion de las caracteristicas, algo que es muy importante realizar para modelos como la Regresion Logistica (Logistic Regression) y SVM (Support Vector Machines), que son sensibles a las escalas de las variables.

```
CODE
```

Por otro lado, el dataset se divide en dos conjuntos: Uno de entrenamiento (0.75) y otro de prueba (0.25). Esta division se hace de tal forma que la proporcion de clases en ambos conjuntos sea similar, evitando asi posibles desbalanceos.

3.4 Regresion Logistica (Logistic Regression)

La Regresion Logistica es un modelo lineal, por lo que es mas adecuado cuando las caracteristicas tienen relaciones lineales con las etiquetas. En este caso, el modelo muestra un buen rendimiento general, aunque la precision podria mejorar utilizando tecnicas de seleccion de caracteristicas o ajustando mas los parametros.

A continuacion, se explican brevemente los parametros utilizados:

- C: El parametro de regularizacion C controla la penalizacion aplicada a los errores. Un valor alto de C indica que se permite una menor penalizacion, por lo que el modelo intenta ajustar mas los datos.
- solver: El solver 'lbfgs' es un optimizador recomendado para solucionar problemas pequeños y medianos.
- multi-class: Procedente de las siglas OnevsRest (OVR), significa que para la clasificacion multiclase, el modelo entrenara un clasificador idenpendiente para cada clase.

```
CODE
```

Se ha obtenido como resultado 165 muestras mal clasificadas, un numero relativamente bajo para este modelo, y se obtuvo una precision de 0.588, por lo que esa proporcion de predicciones del modelo fur correcta.

3.5 **Maquinas de Soporte Vectorial (SVM)**

SVM con kernel RBF es potente en la captura de relaciones no lineales entre las características. El modelo muestra un rendimiento sólido y, aunque los resultados dependen mucho de la selección de los parámetros gamma y C, en este caso el ajuste da buenos resultados.

A continuacion, se explican brevemente los parametros utilizados:

- kernel = 'rbf': Se utiliza el kernel radial base (RBF), que es adecuado para problemas no lineales.
- gamma = 0.7: Controla el grado de influencia de los puntos individuales. Un valor bajo significa que el área de influencia de cada punto es alta, mientras que un valor alto restringe el área.
- C = 30.0: Controla el grado de penalización aplicado a los errores de clasificación. Un valor más alto de C tiende a reducir los errores de clasificación en el conjunto de entrenamiento.

```
CODE
```

Se ha obtenido como resultado 157 muestras mal clasificadas, un numero relativamente bajo para este modelo, y se obtuvo una precision de 0.608, una proporcion ligeramente superior a la obtenida con la Regresion Logistica.

3.6 **Arboles de Decision (Random Trees)**

Los árboles de decisión tienden a sobreajustar los datos si no se limitan adecuadamente. En este caso, al limitar la profundidad a 4, se puede evitar el sobreajuste y el modelo puede ser generalizado adecuadamente. Sin embargo, en comparación con los otros modelos, el rendimiento fue ligeramente inferior, posiblemente porque el modelo no pudo capturar todas las complejidades de los datos con solo 4 niveles de profundidad.

A continuacion, se explican brevemente los parametros utilizados:

- criterion = 'gini': Utiliza el índice de Gini para medir la pureza de los nodos.
- max-depth = 4: La profundidad máxima del árbol se fija en 4 para evitar el sobreajuste.

```
CODE
```

Se ha obtenido como resultado 172 muestras mal clasificadas, un numero similar a los otros dos modelos, y se obtuvo una precision de 0.570, por lo que esa proporcion de predicciones del modelo fur correcta.

4 Clasificación usando 2 características de los datos

4.1 Introduccion

En este apartado se presentan los resultados de la clasificacion del conjunto de datos del fichero dataset.csv utilizando 3 metodos de aprendizaje supervisado: Regresion Logistica (Logistic Regression), Maquinas de Soporte Vectorial (SVM) y Arboles de Decision (Random Trees).

Cada uno de estos clasificadores ha sido entrenado usando todas las caracteristicas del dataset. A continuacion, se iran describiendo los parametros utilizados para cada modelo, presentando los resultados obtenidos y discutiendo el rendimiento de cada clasificador.

4.2 Seleccion de las 2 caracteristicas

En primer lugar, se realizó un análisis de correlación entre las variables del dataset completo para identificar las características más relevantes. Se seleccionaron las siguientes cuatro características basadas en su diversidad y posible relación con la variable objetivo:

Col1 Col5 Col7 Col11

Estas variables se seleccionaron por su correlación baja entre sí y por su posible relevancia para la clasificación, a fin de evitar multicolinealidad y maximizar la capacidad predictiva del modelo.

```
CODE
```

4.3 Preparacion de los datos

Antes de entrenar los modelos, primero se debe cargar el dataset, la anonimizacion de las caracteristicas y la separacion de las caracteristicas (X) respecto a la etiqueta objetivo (y).

Ademas de esto, tambien se realiza una estandarizacion de las caracteristicas, algo que es muy importante realizar para modelos como la Regresion Logistica (Logistic Regression) y SVM (Support Vector Machines), que son sensibles a las escalas de las variables.

```
CODE
```

Por otro lado, el dataset se divide en dos conjuntos: Uno de entrenamiento (0.75) y otro de prueba (0.25). Esta division se hace de tal forma que la proporcion de clases en ambos conjuntos sea similar, evitando asi posibles desbalanceos.

4.4 Regresion Logistica (Logistic Regression)

La Regresion Logistica es un modelo lineal, por lo que es mas adecuado cuando las caracteristicas tienen relaciones lineales con las etiquetas. En este caso, el modelo muestra un buen rendimiento general, aunque la precision podria mejorar utilizando tecnicas de seleccion de caracteristicas o ajustando mas los parametros.

A continuacion, se explican brevemente los parametros utilizados:

- C: El parametro de regularizacion C controla la penalizacion aplicada a los errores. Un valor alto de C indica que se permite una menor penalizacion, por lo que el modelo intenta ajustar mas los datos.
- solver: El solver 'lbfgs' es un optimizador recomendado para solucionar problemas pequeños y medianos.
- multi-class: Procedente de las siglas OnevsRest (OVR), significa que para la clasificacion multiclase, el modelo entrenara un clasificador idenpendiente para cada clase.

```
CODE
```

Se ha obtenido como resultado 165 muestras mal clasificadas, un numero relativamente bajo para este modelo, y se obtuvo una precision de 0.588, por lo que esa proporcion de predicciones del modelo fur correcta.

4.5 Maquinas de Soporte Vectorial (SVM)

SVM con kernel RBF es potente en la captura de relaciones no lineales entre las características. El modelo muestra un rendimiento sólido y, aunque los resultados dependen mucho de la selección de los parámetros gamma y C, en este caso el ajuste da buenos resultados.

A continuacion, se explican brevemente los parametros utilizados:

- kernel = 'rbf': Se utiliza el kernel radial base (RBF), que es adecuado para problemas no lineales.
- gamma = 0.7: Controla el grado de influencia de los puntos individuales. Un valor bajo significa que el área de influencia de cada punto es alta, mientras que un valor alto restringe el área.
- C = 30.0: Controla el grado de penalización aplicado a los errores de clasificación. Un valor más alto de C tiende a reducir los errores de clasificación en el conjunto de entrenamiento.

```
CODE
```

Se ha obtenido como resultado 157 muestras mal clasificadas, un numero relativamente bajo para este modelo, y se obtuvo una precision de 0.608, una proporcion ligeramente superior a la obtenida con la Regresion Logistica.

4.6 Arboles de Decision (Random Trees)

Los árboles de decisión tienden a sobreajustar los datos si no se limitan adecuadamente. En este caso, al limitar la profundidad a 4, se puede evitar el sobreajuste y el modelo puede ser generalizado adecuadamente. Sin embargo, en comparación con los otros modelos, el rendimiento fue ligeramente inferior, posiblemente porque el modelo no pudo capturar todas las complejidades de los datos con solo 4 niveles de profundidad.

A continuacion, se explican brevemente los parametros utilizados:

- criterion = 'gini': Utiliza el índice de Gini para medir la pureza de los nodos.
- max-depth = 4: La profundidad máxima del árbol se fija en 4 para evitar el sobreajuste.

```
CODE
```

Se ha obtenido como resultado 172 muestras mal clasificadas, un numero similar a los otros dos modelos, y se obtuvo una precision de 0.570, por lo que esa proporcion de predicciones del modelo fur correcta.

5 Comparación de los resultados y conclusiones

5.1 **Comparacion de los resultados**

TABLA

El SVM con kernel RBF fue el modelo más consistente en cuanto a rendimiento en todas las configuraciones. La Regresión Logística mostró un rendimiento sólido pero decreciente a medida que se redujeron las características. Los Árboles de Decisión fueron los más afectados por la reducción de información, lo que sugiere que su capacidad para construir reglas de decisión eficaces depende más de la cantidad de características disponibles.

5.2 **Conclusiones**

El rendimiento de los clasificadores depende significativamente del número de características utilizadas. Los hallazgos clave son:

SVM (kernel RBF) fue el más robusto frente a la reducción de características, adaptándose bien incluso con solo dos variables.

La Regresión Logística mostró un rendimiento decreciente cuando se redujo la cantidad de características, destacando su dependencia en información más completa para realizar predicciones precisas.

Árboles de Decisión necesitan más características para crear reglas complejas y precisas; con menos variables, su capacidad para capturar patrones importantes se ve disminuida.

En general, reducir el número de características impacta el rendimiento de los modelos, pero algunos, como el SVM, pueden manejar esta reducción mejor que otros. Para futuros estudios, sería útil explorar técnicas de selección o extracción de características para mejorar el rendimiento mientras se reducen las dimensiones del dataset.