Facultad de Matemática, Astronomía y Física y Computación
Universidad Nacional de Córdoba

Ciencia de Datos

Parcial N° 2 - Junio 4, 2024

Problema 1: [2 pts]
a) ¿Cuáles son las similitudes y diferencias entre Boosting y Bagging?
b) ¿Cuál es el mejor y por qué?
c) Dar un ejemplo de cada uno.
d) Boosting y Bagging, son solamente relevantes en un contexto de árboles de decisión?
Responder en una celda de texto en un colab.

Problema 2: [2 pts] La tabla muestra una salida del clasificador del problema anterior que devuelve un *score* entre 0 y 1 al clasificar cada ejemplo. La clase (C) del problema tiene los valores S (spam) o N (no spam).
a) Construir la correspondiente *curva* ROC.
b) Calcular AUC.
c) Elegir y justificar cuál sería el mejor umbral del *score* para clasificar con mayor `accuracy` este ejemplo.

	C	Score		C	Score
1	S	0.90	6	S	0.40
2	S	0.85	7	N	0.29
3	N	0.71	8	S	0.25
4	S	0.59	9	N	0.20
5	S	0.51	10	N	0.15

Responder en el colab abierto en el problema anterior.

Problema 3: [4 pts] Descargar el Bank Marketing dataset disponible en the UCI Machine Learning Repository. Al descomprimir el archivo se encontrarán a su vez dos archivos.zip y dentro de `bank.zip` se encuentra el archivo `bank-full.csv` de interés para esta tarea.
Los datos se corresponden a una campaña de marketing telefónico directo, implementada por un banco de Portugal para ofrecer a sus clientes la posibilidad de efectuar un depósito de dinero a plazo. La variable clase (`y`) tiene los valores `yes` y `no` según el cliente haya o no suscripto un depósito al finalizar la campaña, respectivamente.
Responder cada uno de los siguientes ítems en celdas separadas en el colab abierto anteriormente. Identificar de forma clara cada ítem del problema en el colab usando sendas celdas de texto.
a) Disponer los datos como dataframe en el colab e indagar el diccionario de atributos. Reportar el número de atributos y ejemplos (clientes) disponibles en el dataset. Reportar el balance de clases, con el número total de ejemplos en cada caso y además los correspondientes porcentajes.
b) Preprocesamiento de los datos:

b1) Identificar los atributos declarados con valores faltantes y eliminar esas columnas.

b2) Identificar los atributos binarios, con valores `yes` y `no` (distintos de la clase) y mapear sus valores a `True` y `False`, respectivamente.

Ayuda: definir `binary_mapper = {'no': False, 'yes': True}` y redefinir cada una de estas columnas usando `df['columna'].replace(binary_mapper)`

b3) Mapear el atributo `month` a números usando

```
df.month = pd.to_datetime(df.month, format='%b').dt.month
```

b4) Identificar los 3 atributos categóricos restantes y eliminar esas columnas.
Verificar el resultado de todas las transformaciones.
c) Rebalancear la clase, manteniendo todos los ejemplos correspondientes a `yes` y reteniendo *al azar* solamente 10000 ejemplos correspondientes a `no`. Reportar el número de ejemplos y atributos del dataframe resultante.
d) Dividir de forma aleatoria los datos para construir un conjunto de entrenamiento, uno para validación y un tercero para test. Reservar el 60 % de los datos para entrenamiento y repartir en partes iguales el resto entre los otros conjuntos. Reportar el tamaño de los tres conjuntos.
e) Usar el modelo `tree.DecisionTreeClassifier()` con `criterion='entropy'` para implementar *cost complexity pruning*. Graficar en función de `alpha` $\in$ [0,0.0025] el número de nodos y la profundidad de los árboles resultantes; y la `accuracy` conseguida en los conjuntos de entrenamiento y validación.
Atención: Este ítem puede tardar más de un minuto en ejecutarse. ¿Cuántos árboles se entrenan?
f) Identificar y reportar el valor de `alpha` para el cual se consigue la mejor `accuracy` en validación. Reportar este valor y la profundidad del correspondiente árbol.
g) Graficar hasta profundidad 3 el árbol del ítem anterior. Usar `fontsize=10` para poder leerlo y reportar los nombre de los atributos que aparecen hasta el nivel 2.
h) Evaluar el árbol del ítem (f) usando el conjunto de test. Mostrar el `classification_report` y la matríz de confusión. ¿Hay indicios de `overfitting`?
i) Mostrar la correspondiente curva ROC y calcular el coeficiente `kappa` de Cohen.
j) Entrenar el modelo `RandomForestClassifier()` usando los siguientes valores de los hiperparámetros: `n_estimators=100, criterion='entropy', max_depth=5, max_features='sqrt'`. Evaluar el modelo sobre el conjunto de test según los ítems (h) e (i). ¿Qué diferencias se encuentran con el árbol del ítem (f)?

Problema 4: [2 pts] En la carpeta `/sample_data` de Google colab se encuentra el archivo `mnist_train_small.csv` con un subset de la MNIST database.
a) Disponer los datos como dataframe. Reportar el número de atributos y ejemplos disponibles. ¿Cómo se explica el número de atributos?
b) Extraer al azar 500 ejemplos, separar la clase (primera columna) y luego aplicar la transformación `MinMaxScaler()` para estandarizar los datos.
c) Usando `n_clusters = [8, 9, 10, 11, 12]` calcular el coeficiente `silhouette` para el modelo `k-means` con el algoritmo de Lloyd, usando los hiperparámetros `init='random', n_init='auto', max_iter=1000` y graficar las siluetas obtenidas. Interpretar el resultado y expresar una conclusión.
d) Fijando el número clusters en 10, evaluar el resultado de `k-means` con los hiperparámetros del ítem anterior, usando los siguientes *scores*: `ajusted Rand index`, `ajusted mutual information`, `homogeneity`, `completeness` y `V measure`.
Nota: En la definición de la función `bench_k_means()`, usar `MinMaxScaler()` de forma de no modificar los datos respecto del ítem anterior.

Descargar la notebook generada en colab y rotular el archivo: `Apellido-parcial02.ipynb` **antes de subirlo a la tarea en el aula virtual. Gracias!**

FaMAF 2024