Trabajo Práctico 1
Clasificación sobre datos simulados

Fecha límite de entrega: 17/10/2018 (23:59hs)
Esqueleto: https://github.com/pbrusco/aa-notebooks

Introducción

Para este trabajo, hemos creado una función generadora de minions. Sobre cada minion, hemos medido 200 características que representan habilidades que poseen en distintas tareas (relacionadas al Mal).

El doctor Nefario ha ideado una fórmula para determinar si un minion es o no apto para concretar su plan para conquistar el mundo. De esta manera ha etiquetado más de 500 minions. Lamentablemente, ha perdido dicha fórmula y necesita seguir decidiendo si nuevos minions son o no aptos para su macabro plan.

Es por esto que nuestro objetivo será construir clasificadores que estimen lo mejor posible la probabilidad de que nuevos minions sean o no aptos para concretar el plan de conquista y así facilitarle las cosas al doctor Nefario.

Por otra parte, ya que el doctor Nefario tuvo problemas con equipos que sobreestiman sus resultados, decidió guardarse varias etiquetas extra que no compartirá con nadie,y que luego utilizará para elegir al mejor equipo, al cual contratará para (de una vez por todas) conquistar el mundo.

En concreto:

Tendrán disponible una matriz de datos ***X*** *de 500 filas en donde cada fila* $\boldsymbol{x}^{(i)}$ representa un vector de 200 características de cada instancia. Es decir, $\boldsymbol{x}^{(i)} = x_1^{(i)}, \ldots, x_{200}^{(i)}$ con *i* entre 1 y 500. Además, tendrán *y*, un vector de 500 posiciones con dos posibles valores: True y False.

Por otra parte, tendrán disponibles más instancias de evaluación ***X_competencia1*** sin las respectivas etiquetas que utilizaremos para evaluar sus resultados.

Ejercicios

Ejercicio 1: Separación de datos

Contarán con una cantidad limitada de datos, por lo cual es importante tomar una buena decisión en el momento de empezar a utilizarlos. En este punto pedimos que evalúen cómo separar sus datos para **desarrollo** y para **evaluación** tomando en cuenta la competencia 1.

Ejercicio 2: Primeros modelos

Para este punto, la tarea consiste en construir y evaluar modelos de tipo árbol de decisión, de manera de obtener una **performance realista** de los mismos.

1. Entrenar un árbol de decisión con profundidad máxima 3 y el resto de los hiperparámetros en default.

2. Calcular la performance del modelo utilizando K-fold cross validation con K = 5, utilizando las métricas “Accuracy” y “ROC AUC”. Para ello, deberán medir la performance en cada partición tanto sobre el fold de validación como sobre los folds utilizados para entrenamiento. Luego, completar la siguiente tabla (directo en el notebook).

Partición	Accuracy (training)	Accuracy (validación)	ROC AUC (training)	ROC AUC (validación)
1				
2				
3				
4				
5				

3. Entrenar árboles de decisión para cada una de las siguientes combinaciones y completar la tabla (completarla directo en el notebook)

Altura máxima	Criterio de evaluación de corte	Accuracy (training)	Accuracy (validación)
3	Gini		
5	Gini		
Infinito	Gini		
3	Ganancia de Información		
5	Ganancia de Información		
Infinito	Ganancia de Información		

EJERCICIO EXTRA: Utilizar la implementación de árboles de decisión que realizaron para la guía de ejercicios. Adaptarlo para que cumpla con la interfaz requerida por sklearn, asegurarse que funcione con variables continuas y reproducir las tablas anteriores.

Ejercicio 3: Comparación de algoritmos

Se pide explorar distintas combinaciones de algoritmos de aprendizaje e hiperparámetros, de manera de encontrar una performance óptima. Para este ejercicio es necesario que evalúen posibilidades utilizando la técnica de Grid Search. Como métrica de performance, usar siempre el área bajo la curva (AUC ROC) resultante de 5-fold cross-validation.

Algoritmos a probar: KNN, árboles de decisión, LDA, Naive Bayes y SVM.

Hiperparámetros: Revisar la documentación de cada uno para la búsqueda de combinaciones prometedoras.

Se pide generar un reporte que contenga:

(a) Cuáles fueron los distintos algoritmos probados, los hiperparámetros considerados y su performance asociada (al menos de los 3 mejores para cada algoritmo).

(b) Una breve explicación de las características de los datos que creen que produjeron dicho resultado.

En este punto evaluaremos tanto los hiperparámetros elegidos como las conclusiones relacionadas a por qué piensan que ciertos algoritmos funcionan mejor que otros para estos datos.

EJERCICIO EXTRA: Utilizar RandomizedSearchCV con rangos de parámetros que contengan a los utilizados en el GridSearch. Analizar si se encontraron mejores combinaciones de parámetros que no hayan sido tenidas en cuenta con el GridSearch y cuál fue la diferencia de tiempo de ejecución.

Ejercicio 4: Diagnóstico Sesgo-Varianza

En este punto, se pide inspeccionar dos de sus mejores modelos: el mejor modelo de tipo árbol de decisión y el mejor de tipo SVM. Para ello:

(a) Graficar curvas de complejidad para cada modelo, variando la profundidad en el caso de árboles, y el hiperparámetro C en el caso de SVM. Diagnosticar cómo afectan al sesgo y a la varianza esos dos hiperparámetros.

(b) Graficar curvas de aprendizaje para cada modelo. En base a estas curvas, sacar conclusiones sobre si los algoritmos parecen haber alcanzado su límite o aumentar la cantidad de datos debería ayudar.

(c) Construir un modelo RandomForest con 200 árboles. Explicar para qué sirve el hiperparámetro *max_features* y explorar cómo afecta a la performance del algoritmo mediante una curva de complejidad. Explicar por qué creen que se dieron los resultados obtenidos. Por último, graficar una curva de aprendizaje sobre los parámetros elegidos para determinar si sería útil o no conseguir más datos (recomendamos grid search para encontrar una buena combinación de parámetros).

Ver: http://scikit-learn.org/stable/modules/learning_curve.html#learning-curve
Atención: Tener en cuenta que debemos seguir utilizando ROC AUC como métrica para estas curvas.

EJERCICIO EXTRA: Utilizar RandomizedSearchCV para explorar la performance del algoritmo de Gradient Boosting y comparar con los resultados obtenidos en el punto (c).

Competencias

Competencia 1: Estimar la performance que tendrá su mejor modelo en datos de evaluación. Recomendamos **no perder de vista** esta competencia en el momento de separar los datos.

Competencia 2: Abriremos una competencia en Kaggle, en donde podrán competir por obtener el mejor modelo posible para estos datos. Estará abierta hasta el final del cuatrimestre.

Entrega

- Contarán con un esqueleto en formato Jupyter Notebook en donde tendrán que completar las celdas faltantes (ya sea con explicaciones y gráficos o código).
- El notebook final deberá ser entregado en formato .html e .ipynb. Es necesario que los resultados puedan reproducirse al ejecutar todas las celdas en orden (Kernel - Restart and Run All)
- Tienen tiempo hasta las 23:59hs del día miércoles **17/10/2018**. La entrega se debe realizar a través del campus virtual y debe contener el informe.
- El trabajo deberá elaborarse en grupos de 3 personas.
- Se podrán pedir pruebas de integridad y autoría; es decir, verificar que la salida solicitada es fruto del modelo presentado y que el modelo fue construido según lo requerido en este enunciado.
- La evaluación será grupal y se basará en la calidad del informe (presentación, claridad, prolijidad); la originalidad, practicidad y coherencia técnica de la solución; la corrección y solidez de las pruebas realizadas.
- En el primer parcial se incluirá una pregunta sobre la solución entregada. Esa pregunta no influirá en la nota del parcial, pero sí en la nota individual del TP1.
- La participación en la competencia 1 es obligatoria y la 2 es opcional. En ambos casos, los resultados no incidirán en la nota de la materia.
- Los ejercicios extra son opcionales para aprobar el TP, pero son obligatorios para **promocionar** la materia.