Universidad del País Vasco Euskal Herriko Unibertsitatea

Sistemas de Ayuda a la Decisión

Práctica 3

Índice

1. Objetivos y Descripción de los contenidos

Esta práctica tiene como objetivo adquirir las competencias que serán evaluadas en la segunda prueba práctica individual. Por lo tanto esta tarea se enmarca dentro de la evaluación continua pero no dentro de los hitos evaluables.

Las **competencias** que el alumno deberá haber adquirido tras realizar la práctica son:

1. Capacidad para desarrollar prototipos de clasificación empleando dos algoritmos: KNN y Arboles de Decisión en Dataiku.
2. Capacidad para a partir de ese prototipo básico de Dataiku exportarlo a un Notebook y a partir de ahi, generar un programa en Python para el clasificador KNN y otro para el Arbol de Decisión.
3. Capacidad para realizar un barrido de hiperparámetros.
4. Capacidad para evaluar la bonanza de caa modelo y seleccionar el mejor.
5. Capacidad para dado un conjunto de instancias *nuevas* se clasifiquen estas empleando el mejor modelo seleccionado.

2. Materiales disponibles

El siguiente material será empleado para el desarrollo de las tareas que se proponen y que tiene como objetivo alcanzar las competencias enumeradas anteriormente.

1. Datos: Se podrán encontrar en eGela dos juegos de datos (iris.csv y SantanderTraHalfHalf.csv). Con cada uno se realizará una experimentación completa.
2. Transparencias sobre los algoritmos KNN y Arboles de Decisión.
3. Cheatsheet de pandas https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf
4. Pantilla de un prototipo en Python para comparar con la exportación de Dataiku.
5. Se os recomienda instalar Anaconda. Documento en eGela[1]
6. Instalar sklearn: pip install scikit-learn o preferiblemente a través de Anaconda.
7. Si se quiere instalar con conda por línea de comando.
8. Instalar imblearn: pip install imblearn.
9. instalar pickle: pip install pickle5

3. Tareas

Dataiku permite obtener modelos de manera rápida para una combinación concreta. Unos datos, un algoritmo y unos hiperparámetros. Pero si queremos hacer un barrido de hiperparámetros o ejecutar en background Dataiku comienza a mostrar sus limitaciones. Este laboratorio pretende que aprendáis a realizar vuestra experimentación en python. Para obtener los objetivos buscados en este proyecto se proponen las siguientes **tareas**:

1. Experimento KNN:
   - Obtener un primer prototipo de KNN empleando Dataiku.
   - Exportar dicho prototipo a un Notebook de Python y comparalo con las plantillas python (scripts) que se os proporcionan. Este script lo tenéis que personalizar, modificandolo hasta hacerlo vuestro y transformarlo para que muchos de los detalles que están hardcoded (por ejemplo los atributos, el preproceso que se le aplica a cada uno de ellos,...) no lo estén. Vuestro scripts debe ser lo más general y automatizado posible y debe responder a las preguntas que se plantean a lo largo de este documento.

[1] Con un enviroment de 3.7 tanto imblearn como pickle5 están en los repositorios y se pueden instalar con Anaconda
Con 3.9 pickle viene preinstalado así que no lo encuentra porque no es necesario instalarlo

- Ejecutar el Notebook paso a paso entendiendo cada instrucción que realiza. Pare ello quizás debáis consultar la documentación de pandas (https://pandas.pydata.org/docs/user_guide/index.html) y de python dependiendo de la versión (https://docs.python.org/es/3.7/,https://docs.python.org/es/3.8/,https://docs.python.org/es/3.9/).
- Sustituir todas las referencias que hubiera a la librería Dataiku, y emplear la librería scikit-learn (Sklearn) de Python (https://scikit-learn.org) y imblearn (https://imbalanced-learn.org/stable/).
- Comparar con las plantillas de Python que se proporciona (Nota: hay dos plantillas una para entrenar y otra para predecir dadas nuevas instancias y el modelo entrenado y almacenado con pickle).
- Añadir comentarios en la plantilla Python explicado cada paso. Modificar los comentarios *# Explica lo que se hace en este paso* por una explicación.
  - ¿Cómo generalizo para cualquier atributo no solo para los de los datos de la plantilla? (pista, emplear el método columns de los dataframes de pandas.https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.columns.html).
  - ¿Cómo accedo a determinados atributos de un dataframe? (Consultar:https://www.activestate.com/resources/quick-reads/how-to-access-a-column-in-a-dataframe-using-pandas/).
  - ¿Cómo selecciono si quiero o no preprocesar los datos? ¿A través de los argumentos del script de entrenamiento por ejemplo? (mirar en el script la parte referente a argv que se os ha proporcionado. Si queréis profundizar: https://machinelearningmastery.com/command-line-arguments-for-your-python-script/)
  - Para seleccionar decisiones referentes al preproceso lo más práctico es generar un configuration.csv donde almacenaré por cada atributo la información que responde a las siguiente cuestiones
    a) ¿Cómo selecciono y elimino atributos si fuese necesario?
    b) ¿Cómo selecciono qué valor (mean, median, etc) para los valores faltantes?

atributo	eliminar	valores faltantes	estandarizacion
atrib1	0	MEAN	Z_SCORE
atrib2	0	MODE	Z_SCORE
...	...	...	...

Cuadro 1: Ejemplo de fichero de configuración config.csv

  - Cuando calculo el valor de la media lo hago sobre los datos del entrenamiento y luego los aplico al conjunto de desarrollo o calculo para cada conjunto sus valores?

  - ¿Voy a querer realizar siempre los mismos preprocesos?
  - ¿Deberé realizar siempre los mismos preprocesos a los datos del test?
- A partir de esa plantilla generar un barrido de parámetros: k:1,3,5; d:1,2;. weights = 'uniform' o weights = 'distance'. Para ello si lo hacéis con python, conviene que añadáis 2 parametros más. K y P. Así tendremos k y K y p y P. k almacenará el valor del menor k y K el mayor valor de K para el barrido de hiper parámetros. Lo mismo con la p y P. p almacenará el menor valor y P el mayor. También conviene añadir otro parámetro -c para marcar que vamos a emplear un fichero de configuración donde almacenaremos como queremos preprocesar cada atributo.
- Evaluar todos los modelos y generar automáticamente un .csv con las figuras de mérito Accuracy, Precision, Recall y F-score para todas las combinaciones de hiperparámetros.

Combinación	Precisión	Recall	F_score(Mac/Mic/Avg/None)
k=5,p=1,uniform	0.XX	0.XX	0.XX
k=5,p=2,uniform	0.XX	0.XX	0.XX
...	...	...	...

- Generar un programa .py o un bash script de forma que se automatice el barrido de hiperparámetros. Tomará como parámetros la k mínima y la k máxima y la p máxima y se encargará de generar todas las combinaciones y con cada combinación invocar al .py que tiene como parámetros un valor de k y un valor de p.
- Una vez hecho el barrido de parámetros, seleccionar el modelo que mejores resultados obtenga. Normalmente es difícil comparar modelos en base a precisión, recall, porque suele pasar que el que es mejor en recall, es peor en precisión, y la accuracy como vimos suele estar sesgada hacia la clase mayoritaria. Así pues, la figura de mérito que se suele emplear es el f1_score (o f_score) que es una media entre la precisión y el recall. Sklearn nos permite obtener esta figura de mérito invocando: f1_score(y_true, y_pred)
  - Si estamos realizando una tarea de clasificación binaria: f1_score(y_true, y_pred average=None), porque solo tenemos dos clases.
  - Si estamos realizando una clasificación multiclase (en el ejemplo del iris):
    - Macro: f1_score(y_true, y_pred, average='macro')
    - Micro: f1_score(y_true, y_pred, average='micro')
    - Weighted: f1_score(y_true, y_pred, average='weighted')

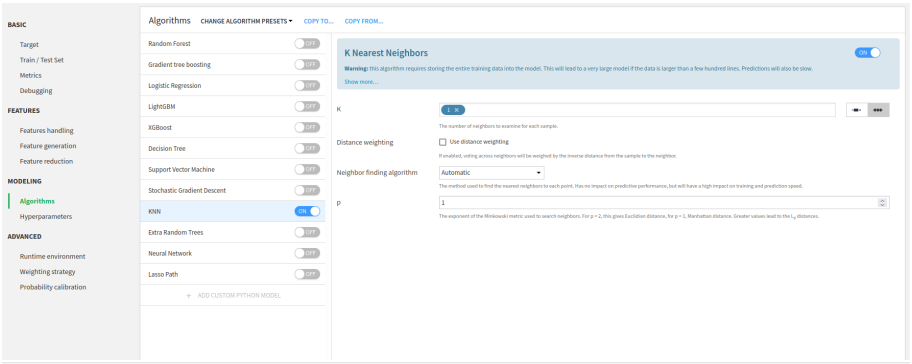
    Consultar la documentación: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

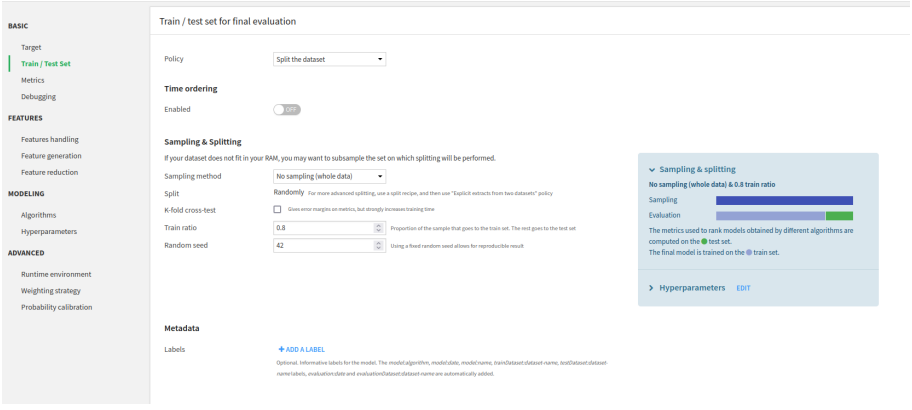
Consultar los apuntes de clase sobre figuras de mérito.

- Regenerar el mejor modelo según la figura de mérito seleccionada (volver a ejecutar el programa que genera el modelo pero con los hiperparametros que mejores resultados nos han dado y salvarlo empleando pickle (Consultar al final del documento).
- Dada una nueva muestra con nuevas instancias que no están clasificadas, obtener la clasificación de las mismas empleando el modelo salvado (Consultar al final del documento).
- Consultar la información sobre la configuración del experimento en Dataiku al final del documento.

2. Experimento Arboles de decisión:

   - Realizar los mismos pasos pero esta vez seleccionando como algoritmo los Arboles de Decisión (https://scikit-learn.org/stable/modules/tree.html#classification).
   - Como hiperparámetros se probará: max_depth=3,6,9,número máximo de atributos que Dataiku considere relevantes combinado con min_samples_split o min_samples_leaf=1 y 2. Es decir, 12 combinaciones como mínimo (siempre sois libres de ampliar el barrido).





4. Receta para salvar modelos

Receta para salvar un modelo en disco:

import pickle

nombreModel = "nombreParAlmacenar.sav"

saved_model = pickle.dump(**clf**, open(nombreModel,'wb')) **clf** o como se llame la clase que contiene el modelo

print('Modelo guardado correctamente empleando Pickle')

Receta para recargar un modelo del disco para poder clasificar nuevas instancias:

import pickle

X_nuevo = pd.read_csv(iFile) contendrá instancias nuevas sin la clase p.q. eso es lo que queremos predecir

nombreModel = "nombreParAlmacenar.sav"

clf = pickle.load(open(nombreModel, 'rb'))

resultado = clf.predict(X_nuevo)

Referencias

[1] Dataset Santander Customer Satisfaction: Solo se ha empleado una parte de estos datos. Es importante decir que se emplearán exclusivamente para hacer la práctica y que el alumno se compromete a no ponerlos en su github. Si el alumno quisiera poner su código en lo referente a los datos deberá nombrar la fuente de los datos para facilitar su replicabilidad, pero no podrá subir los datos dado que su uso está restringido:

'Data' means the Data or Datasets linked from the Competition Website for the purpose of use by Participants in the Competition. For the avoidance of doubt, Data is deemed for the purpose of these Competition Rules to include any prototype or executable code provided to Participants by Kaggle or Competition Sponsor via the Website. Participants must use the Data only as permitted by these Competition Rules and any associated data use rules specified on the Competition Website.

Unless otherwise permitted by the terms of the Competition Website, Participants must use the Data solely for the purpose and duration of the Competition, including but not limited to reading and learning from the Data, analyzing the Data, modifying the Data and generally preparing your Submission and any underlying models and participating in forum discussions on the Website. Participants agree to use suitable measures to prevent persons who have not formally agreed to these Competition Rules from gaining access to

the Data and agree not to transmit, duplicate, publish, redistribute or otherwise provide or make available the Data to any party not participating in the Competition. Participants agree to notify Kaggle immediately upon learning of any possible unauthorized transmission or unauthorized access of the Data and agree to work with Kaggle to rectify any unauthorized transmission. Participants agree that participation in the Competition shall not be construed as having or being granted a license (expressly, by implication, estoppel, or otherwise) under, or any right of ownership in, any of the Data.

[2] Iris Dataset UCI Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/iris

[3] Competencias asociadas de la tarea: Se han tomado como referencia los apuntes de SAD 2020-2021 (fuente: Alicia Pérez) para así coordinar que la práctica y las competencias a obtener sea lo más similar posible a lo solicitado en años anteriores.