Solución del reto filtro

Equipo: *Compu mundo hyper mega red*

Integrantes:

1. Santiago de Jesús González Medellín
2. Noé Hoyos Quiroz
3. Pedro Romero Martinez

Repositorio de soluciones:

Aquí

Descripción de datos

Los datos de inicio que se declararon durante la solución del problema *reto filtro* son: * `datosEntrenamiento.csv`: Archivo requerido y obligatorio donde contiene datos numericos clasificados las filas por clases. * `datosPrueba.csv`: Archivo requerido y obligatorio donde contiene datos numericos clasificados las filas por clases. * `datosValidacion.csv`: Archivo que contiene la validación que corresponte a cada clase de los datos `datosEntrenamiento.csv` y `datosPrueba.csv`. * `retoFiltro.txt`: Archivo donde debe mostrar el resultado generado por el algoritmo, mostrando a que clase pertenece cada uno de los objetos de la colección de datos que se encuentra en el archivo `datosValidacion.csv`, aplicando la metodología empleada para generar el modelo.

Justificación de modelos

Nosotros decidimos atacar el problema utilizando modelos de *machine learning*, después de explorar los datos y teniendo en cuenta que la naturaleza de las salidas es discreta, optamos por utilizar aprendizaje supervisado, en particular modelos de clasificación.

Exploración de modelos

Intentamos el problema de clasificación con el lenguaje de programación `python 3.6` utilizando las siguientes librerias:

1. `pandas`: Para lectura y preprocesamiento de los datos de entrenamiento y de prueba.
2. `numpy`: Para preprocesamiento de los datos de entrenamiento y de prueba.
3. `scikitlearn`: Obtener instancias de los modelos, entrenarlos y clasificar datos de prueba y validación.

Además utilizamos `jupyter notebook` para poner el código de cada uno de los modelos en notebooks.

Los modelos de clasificación que utilizamos fueron *decision trees* y *multi layer perceptron network*, ambos modelos están incluidos en `scikitlearn` con las funciones `DecisionTreeClassifier()` y `MLPC()` respectivamente.

El primero intento fue el modelo de *decision trees*, el codigo esta en './filter/pedro_solutions/filterSolTree.ipynb' en el repositorio. Este modelo fue entrenado con los datos de entrenamiento proporcionados. Antes de entrenar el modelo se cálculo la correlación de Pearson entre las columnas de los datos, se observaron altas correlaciones y se eliminaron algunas columnas, de esta manera se obtuvo una precision aproximadamente del 83% sobre los datos de prueba y despues utilizando todas las columnas se obtuvo una precision aproximadamente del 85% sobre los datos de prueba.

Finalmente el otro modelo utilizado fue *multi layer perceptron network*, que se explica en la siguiente sección, con el que se obtuvo una precisión de aproximadamente el 90%, esto se explica más a detalle a continuación.

Modelo elegído

Se implentó una red de perceptrones multicapa(MLPN por sus siglas en inglés) para la tarea de clasificación de los datos proporcionados. El motivo de esto es que se ha probado que las redes neuronales, cuando están bien configuradas, pueden obtener buenos resultados. Para la tarea del reto filtro se implementó un clasificador de esta clase. El código se ubica en './filter/santigm/MLPN.ipynb' dentro del repositorio.

Las redes neuronales tienen la capacidad de aprender modelos no lineales por lo que son adecuadas para esta tarea ya que, a pesar de tener muchos atributos con un alto índice de correlación(>90%), existen atributos que no presentan una relación lineal entre ellos.

Para entrenar al modelo se utiliza la siguiente metodología.

- `Escalamiento de datos`. Se escalan los atributos (del 1 al 36) de los archivos de entrenamiento y prueba en el rango [0,1) puesto que las MLPN's trabajan mejor dentro de ese rango.
- `Codificación de variable objetivo`. La variable objetivo (atributo 37) se transforma en códigos (etiquetas) que no inducen ningún orden y que las librerías que se utilizaron pueden trabajar.
- `Selección de modelo`. Se realiza una validación cruzada(cross-validation) de 10 secciones para ajustar parámetros que se adecuan mejor a nuestro modelo neuronal. La selección se hace en base a la mejor precisión promedio que se obtuvo de cada una de las particiones y la varianza de la presición.
- `Entrenamiento del modelo`. Con los parámetros que exhibieron el mejor desempeño se entrenó el modelo con los datos de entrenamiento.
- `Prueba del modelo`. Una vez entrenado se evalua el modelo con los datos de prueba. Con este conjunto de datos se obtiene una presición del 90.1%.
- `Generación del archivo de respuesta a los datos de validación(retoFiltro.txt)`. Con el modelo entrenado y probado se genera el archivo de respuesta para el reto filtro y se guarda el modelo.

Conclusiones

Los datos del reto filtro fueron fáciles de trabajar puesto que no requirieron de un

preprocesamiento tan elaborado. Los modelos que se probaron son especificamente para tareas de aprendizaje supervisado, en específico para tareas de clasificación. Los resultados del entrenamiento fueron aceptables posiblemente debido a la metodología de implementación.

Referencias

<Link> : http://scikit-learn.org/stable/modules/neural_networks_supervised.html#neural-networks-supervised

<E-book> : Data Science from Scratch - *Joel Grus* - 2015 - ISBM: 978-491-90142-7