Equipo: ETSIIT++

Análisis exploratorio de los datos y comprensión de negocio. Manipulación de variables y su argumentación: eliminación, transformación y/o creación de nuevas variables

Tenemos un conjunto de clientes, que ha hecho una serie de operaciones para adquirir diferentes productos. En el dataset que nos entregan, figuran cada una de esas operaciones.

El problema a resolver sería por tanto, dado un conjunto de operaciones de clientes, adivinar cuál será el próximo producto que contratará. La representación del problema, será la siguiente:

Variables independientes:

- Variables Socio_DemoN [0-5]
- 94 variables binarias, que indican si el cliente ya tiene un determinado producto o no.

Variable dependiente:

- Próximo producto que contratará (categórica)

Hemos planteado este problema como un problema de clasificación, multiclase, en la que usaremos como target el próximo producto a comprar por el usuario.

Generación de Datasets de Train/Test

Para cada registro de contratación, podemos crear un “caso” de nuestro dataset. La columna “Cod_Prod” pasará a ser el target. Además incorporaremos 94 columnas nuevas, para expresar de forma binaria los productos que el cliente ya tiene.

El proceso en python para convertir el dataset de operaciones en el dataset que necesitamos puede encontrarse en preprocessing/preprocessing.py

Selección de modelo/s y ajuste de parámetros. Elección de métrica elegida y justificación. Bondad del modelo. Interpretabilidad del modelo y viabilidad de puesta en producción

Métrica elegida

En todos los algoritmos usados tenemos las mismas métricas:

- Precisión: Para saber qué porcentaje de aciertos tiene nuestro algoritmo sobre el total de los productos predichos.
- Coeficiente Kappa: Mide la concordancia entre valores reales y predichos, ajustada al azar.

Para evaluar los algoritmos y su precisión, se ha hecho una simulación, se han separado, la última operación de cada clientes del resto de operaciones.

Por tanto, con este set de validación, estaríamos midiendo la precisión de este algoritmo con la última operación de cada cliente, como una aproximación de la precisión que se debería alcanzar en una futura, nueva, operación.

Ensemble RandomForest, ExtraTrees y XGBoost

Para ambos hemos usado una selección de características basada en entrenar un ExtraTrees con todas las columnas del dataset y quedarnos con las 40 mejores:

```
cols = [u'SD1', u'SD2', u'SD3', u'SD4', u'SD5', u'P4', u'P5', u'P10', u'P11',
       u'P13', u'P15', u'P18', u'P21', u'P30', u'P37', u'P45', u'P48', u'P49',
       u'P54', u'P57', u'P60', u'P64', u'P69', u'P71', u'P72', u'P73', u'P74',
       u'P76', u'P77', u'P78', u'P79', u'P80', u'P82', u'P83', u'P85', u'P90',
       u'P91', u'P92', u'P93']
```

El código para la selección de características puede encontrarse en models/features_selection.py.

En los 3 ensembles los resultados han sido similares, comportándose algo mejor XGBoost. Se puede consultar los resultados en la Tabla 1, adjunta abajo.

El código de los ensembles puede encontrarse en models/et_ensemble.py, models/rf_ensemble.py y models/xgb_ensemble.py. La parametrización ha sido escogida después de hacer diferentes pruebas hasta encontrar valores que daban buenos resultados.

Modelo	Accuracy	Kappa
ExtraTrees	48,61 %	43,5 %
RandomForest	48,77 %	43,8 %
XGBoost	50.88 %	46.06 %

Tabla 1: Resultados ofrecidos por cada modelo

Legibilidad, estructura y reproducibilidad del código:

En el archivo .zip enviado podrán encontrar diferentes archivos y carpetas necesarias para la reproducción del código.

Recomendamos usar python en su versión 2.7. Concretamente nosotros hemos usado 2.7.12.

El resto de paquetes necesarios se encuentran en requirements.txt, y recomendamos instalarlos usando pip:

```
$ pip install -r requirements.txt
```

Si se quiere usar el archivo xgb_ensemble.py donde se usa el XGBoost, es necesario no instalar con un simple “pip install xgboost”, porque se han usado características recientemente incluidas en el proyecto. Para más info:
https://github.com/dmlc/xgboost/issues/1950

Por tanto, los pasos a seguir para instalar XGBoost serían:

```
$ git clone https://github.com/dmlc/xgboost
$ cd xgboost/python-package
$ sudo python setup.py install
$ export PYTHONPATH=~/<your-path>/xgboost/python-package
```

En la carpeta principal encontrarán 4 carpetas:

- **preprocessing:** Hay scripts necesarios para el procesamiento de los dataset que ofrece Cajamar.
- **edas:** Hay diferentes Análisis Exploratorios de Datos.
- **models:** Se encuentra la implementación de los modelos comentados en este mismo documento.
- **data:** Aquí deberían encontrarse los ficheros (train2.txt y test2.txt) que provee Cajamar para el reto. No los hemos adjuntado porque son descargables desde la web de Cajamar.

Posibles mejoras futuras:

- Estudiar con más detalle las secuencias de compras que muestra los clientes, aunque esto se recoge en el preprocesado llevado a cabo, creemos que con Redes Neuronales Recurrentes (RNN, LSTM o GRU) podemos obtener resultados mejores.