FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2

Análisis de Datos FiuMark

Segundo Cuatrimestre 2020

Fecha de Entrega: 16/02/2021

Integrantes:

Nombre y Apellido	Padrón
Anarella Nicoletta	94.551
Juan Cruz Opizzi	99.807

Introducción

El objetivo del presente informe es mostrar los resultados obtenidos al aplicar distintas técnicas de Machine Learning, sobre el set de datos provisto por FiuMark.

Tomando como punto de partida, el análisis exploratorio previo y el baseline definido para el set inicial, se aplicaron diferentes algoritmos, tanto de preprocesamiento como de predicción de datos, a fin de mejorar las predicciones.

Se buscó predecir si la persona que fue al cine a ver la película Frozen 2 regresará al mismo para ver su secuela, Frozen 3. Los datos sobre los espectadores fueron proporcionados por FiuMark, y la evaluación final de los modelos se realizó generando predicciones sobre registros para los que desconocemos si el espectador regresaría o no.

Modelos de Machine Learning

Preprocesamientos utilizados

A continuación detallamos las distintas funciones de preprocesamiento desarrolladas. Las mismas se encuentran en el archivo preprocessing.py y son reutilizadas por distintos modelos.

Nombre	Descripción	Función de Python
Drop features irrelevantes	Elimina los features ‘id_ticket' y 'nombre'. Si drop_fila es True, también elimina 'fila'. (Por defecto vale False)	eliminar_features_que_no_aportan_info (df, drop_fila)
Acompañantes	Genera el feature ‘cant_acompañantes', como la suma de 'parientes' y 'amigos'	crear_feature_acompañantes(df)
Reemplazo nulos (feature: métrica)	Si la columna existe en el DataFrame, reemplaza los valores nulos de la columna con la métrica indicada (que puede ser media, mediana o moda)	replace_nulls_column(df, columna, metrica)
One Hot Encoding (columnas)	Aplicamos One Hot Encoding a la lista de columnas recibida por parámetro. Se usan N-1 filas, siendo N el número de categorías a encodear	encodear_atributos_categoricos(df, columns)
Normalización	Normaliza todos los atributos numéricos del DataFrame	normalizar_atributos_numericos(df)
Drop features (features)	Recibe una lista de features y los elimina del DataFrame	eliminar_features(df, feature_list)
KNN para missing values	Aplica KNN para completar los valores faltantes del dataset	usar_knn_para_missing_values(df)

Modelos utilizados

Hemos aplicado y testeado diferentes modelos de Machine Learning, cuyos resultados procederemos a detallar. Cada modelo funciona con 2 variantes de preprocesado distintas, detallándose en la primera fila el preprocesado 1 y en la segunda fila el preprocesado 2, para cada modelo.

Antes de pasar a los resultados obtenidos nos gustaría hacer algunos comentarios.

En el tunning de los modelos, utilizamos Random Search, para encontrar las mejores combinaciones de hiper-parámetros. Además, al momento de entrenar (con esos hiper-parámetros obtenidos) aplicamos Stratified K-Fold.

En el caso de KNN y el árbol de decisión, usamos Grid Search para encontrar los hiper-parámetros. En el caso de KNN, el limitar las combinaciones posibles al subset que nos interesaba fue lo que nos permitió aplicar el algoritmo. Con respecto al árbol de decisión, como es un modelo simple, rápido y con pocos hiper-parámetros, lo aprovechamos como baseline. Es decir, nos permitió analizar si los modelos más complejos eran demasiado complejos para el caso, o si directamente estaban prediciendo muy mal.

Las métricas utilizadas para evaluar los modelos fueron:

- AUC-ROC
- Matriz de confusión
- Accuracy
- Precisión
- Recall

Finalmente, generamos un archivo aux.py compilando varias funciones que se utilizaron para calibrar y medir la performance de los modelos.

Nombre	Preprocesamientos	AUC-ROC	Matriz de confusión	Accuracy	Precisión	Recall	F1 Score
KNN	Drop features irrelevantes Acompañantes Reemplazo nulos (edad: media) Normalización One Hot Encoding (nombre_sede, genero, tipo_de_sala, fila)	0.9845	TP:70 FP:0 TN:124 FN:7	0.9652	1.0000	1.0000	0.9524
	Drop features irrelevantes Acompañantes Drop features (amigos, parientes) Reemplazo nulos (edad: media) One Hot Encoding (nombre_sede, genero, tipo_de_sala, fila)	0.9727	TP:68 FP:7 TN:117 FN:9	0.9204	0.9067	0.9435	0.8947
Support Vector Classifier (SVC)	Drop features irrelevantes Reemplazo nulos (edad: media) Normalización One Hot Encoding (fila, tipo_de_sala, nombre_sede, genero)	0.8567	TP:41 FP:3 TN:121 FN:36	0.8060	0.9318	0.9758	0.6777
	Drop features irrelevantes Acompañantes Drop features (amigos, parientes, fila) Reemplazo nulos (edad: media) Normalización One Hot Encoding (tipo_de_sala, nombre_sede, genero)	0.8205	TP:47 FP:10 TN:114 FN:30	0.8010	0.8246	0.9194	0.7015
Árbol de decisión	Drop features irrelevantes Drop features (fila)	0.8906	TP:57 FP:12 TN:115 FN:17	0.8557	0.8261	0.9055	0.7972

	Acompañantes Reemplazo nulos (edad: media) Normalización One Hot Encoding (tipo_de_sala, nombre_sede, genero)						
	Drop features irrelevantes One Hot Encoding (tipo_de_sala, nombre_sede, genero, fila) KNN para missing values	0.9122	TP:56 FP:10 TN:117 FN:18	0.8607	0.8485	0.9213	0.8000
Random Forest	Drop features irrelevantes Drop features (fila) Acompañantes Reemplazo nulos (edad: media) One Hot Encoding (tipo_de_sala, nombre_sede, genero)	0.8500	TP:46 FP:21 TN:103 FN:31	0.7413	0.6866	0.8306	0.6389
	Drop features irrelevantes Reemplazo nulos (edad: mediana) One Hot Encoding (tipo_de_sala, nombre_sede, genero, fila) Normalización	0.8479	TP:53 FP:21 TN:103 FN:24	0.7761	0.7162	0.8306	0.7020
Red neuronal	Drop features irrelevantes Drop features (fila) Acompañantes Reemplazo nulos (edad: media) Normalización One Hot Encoding (tipo_de_sala, nombre_sede, genero)	0.8374	TP:39 FP:6 TN:93 FN:23	0.8199	0.8667	0.9394	0.7299
	Drop features irrelevantes Drop features (fila)	0.8409	TP:39 FP:8 TN:91 FN:23	0.8075	0.8298	0.9192	0.7156

	Normalización One Hot Encoding (tipo_de_sala, nombre_sede, genero) KNN para missing values						
Bagging	Drop features irrelevantes Drop features (fila, precio_ticket) Reemplazo nulos (edad: mediana) One Hot Encoding (tipo_de_sala, nombre_sede, genero)	0.8027	TP:41 FP:17 TN:107 FN:36	0.7811	1.0000	1.0000	0.6000
	Drop features irrelevantes Acompañantes Drop features (fila, parientes) Reemplazo nulos (edad: mediana) One Hot Encoding (tipo_de_sala, nombre_sede, genero) Normalización	0.8255	TP:40 FP:17 TN:107 FN:37	0.7313	0.7018	0.8629	0.5970
AdaBoost	Drop features irrelevantes Drop features (fila) Acompañantes Reemplazo nulos (edad: media) Normalización One Hot Encoding (tipo_de_sala, nombre_sede, genero)	0.9062	TP:57 FP:8 TN:116 FN:20	0.8607	0.8769	0.9355	0.8028
	Drop features irrelevantes Reemplazo nulos (edad: mediana) Drop features (fila, amigos, parientes) One Hot Encoding (tipo_de_sala, nombre_sede, genero)	0.8846	TP:53 FP:9 TN:115 FN:24	0.8358	0.8548	0.9274	0.7626

GradientBoost	Drop features irrelevantes Acompañantes Reemplazo nulos (edad: media) Drop features (fila) One Hot Encoding (tipo_de_sala, nombre_sede, genero)	0.9623	TP:63 FP:3 TN:121 FN:14	0.9154	0.9545	0.9758	0.8811
	Drop features irrelevantes Acompañantes Reemplazo nulos (edad: media) Drop features (amigos) One Hot Encoding (fila, tipo_de_sala, nombre_sede, genero)	0.8791	TP:50 FP:9 TN:115 FN:27	0.8209	0.8475	0.9274	0.7353
Voting	Drop features irrelevantes Acompañantes Reemplazo nulos (edad: moda) One Hot Encoding (tipo_de_sala, nombre_sede, genero, fila)	0.8700	TP:42 FP:8 TN:116 FN:35	0.7861	0.8400	0.9355	0.6614
	Drop features irrelevantes Drop features (fila, amigos, parientes) Reemplazo nulos (edad: moda) One Hot Encoding (tipo_de_sala, nombre_sede, genero) Normalización	0.8757	TP:48 FP:9 TN:115 FN:29	0.8109	0.8421	0.9274	0.7164

Conclusiones

En base al análisis y comparaciones realizadas, y teniendo en cuenta que la métrica requerida es el AUC ROC, recomendamos a la empresa FiuMark utilizar el modelo de datos KNN para predecir el regreso de su audiencia.

Este clasificador fue el que obtuvo el mejor score en las métricas evaluadas, pero a la vez sin caer en el caso de overfitting (o memorizado de datos). El hecho de complementar al modelo con acciones de preprocesado que incluyan normalización de los datos, y a la vez potenciar el entrenamiento a través de Stratified K-Fold, hicieron que mejore considerablemente la performance general de KNN.

En el análisis exploratorio previo, habíamos obtenido un accuracy de 0.82, sin aplicar ninguna inteligencia automática de predicción de datos. Considerando que los valores de AUC ROC en KNN son de 0.97 y 0.98 (según el preprocesado aplicado), puede verse claramente una gran mejoría en los resultados obtenidos. Al ser un problema que no presenta mucha complejidad, KNN da muy buenos resultados y supera a la red neuronal.

Es destacable también el resultado obtenido con el preprocesado 1 de GradientBoosting, que presenta un AUC ROC de 0.96. Si por algún motivo no se pudiese aplicar el modelo KNN, éste sin dudas sería la siguiente alternativa a considerar. Pero hay que considerar que la complejidad computacional al utilizar este tipo de modelos es mayor.

En caso de que el objetivo de la empresa FiuMark sea, por ejemplo, ejecutar una acción de Marketing dirigida a la audiencia que tiene intenciones de volver (como ofrecerles un descuento en su próxima entrada), entonces recomendaríamos el modelo que tenga mayor recall. De esta manera, se minimizaría el número de FN y FiuMark se aseguraría de solo dar este beneficio a aquellos espectadores que le son fieles.

Asimismo, el baseline de la primera parte actuó como nuestro benchmark, para determinar si nuestros modelos estaban realizando una buena predicción o no, ya que si un modelo de Machine Learning no posee un score superior al que ofrecen una serie de if's anidados, entonces estamos frente a un problema. El mismo puede estar alocado en el feature engineering, en el entrenamiento, o mismo en el set de hiperparametros, entre otros.

Respecto a por qué creemos que algunos modelos dan mejor que otros:

Como primer vistazo podemos concluir que los mejores modelos son

- Los que pudieron permitirse obtener el mejor conjunto de hiper parámetros que mejor ajusta al set de datos.
- Son a su vez, los modelos más sencillos.

Teniendo en cuenta esto, tiene sentido para nosotros pensar que, para este set de datos se ajusta mejor con un modelo sencillo que uno complejo, es decir el set de datos no es tan complicado como para requerir un modelo muy complejo. Lo que explicaría que Random Forest, Bagging y la red neuronal den relativamente mal. Por otro lado, en una segunda mirada, podemos ver que GradientBosting dio muy bien y regular para 2 subconjuntos distintos de pre procesados, por lo que pensamos que no usamos el mejor pre procesado posible para más de un modelo.

En resumen concluimos que:

- Los modelos sencillos andan muy bien
- Los complejos anda relativamente mal excepto en ocasiones donde el preprocesado es ideal