Facultad de ingeniería | Universidad de Buenos Aires

75.06 | Organización de Datos

Trabajo Práctico
2da Parte

Cátedra Collinet

Primer cuatrimestre de 2021

Alumnos	**Roussilian Juan Cruz**	**104269**
	Stancanelli Guillermo José	**104244**

Introducción:

En el presente informe se presenta un análisis realizado sobre la segunda parte del trabajo práctico, el cual consistía en crear distintos modelos de análisis supervisado para predecir la lluvia de hamburguesas ocasionada por la máquina del Flint, según distintos parámetros climáticos.

Objetivos:

Los objetivos del trabajo práctico son:

- Comprender los modelos de machine learning enseñados a lo largo del cuatrimestre.
- Evaluar y comparar modelos según diferentes métricas, midiendo su funcionamiento.
- Aplicar diferentes preprocesamientos de datos para preparar la información antes del análisis.

Formato de la Entrega:

Se presentan en el repositorio de github otorgado por la cátedra, los siguientes archivos:

1. Jupyter Notebooks:
    - arbol_clasificacion_ipynb
    - Boosting.ipynb
    - knn.ipynb
    - naive_bayes.ipynb
    - RandomForest.ipynb
    - red_neuronal.ipynb
    - svm.ipynb

2. Archivos .py:
    - preprocessing.py : Contiene los preprocesamientos utilizados en los modelos.
    - funciones_auxiliares.py : Funcionalidad de rutina utilizada en los notebooks.
    - requirements.txt: Contiene las dependencias para ejecutar el proyecto.
    - /predicciones : Contiene los exports a .csv de todos los modelos, según el dataset de predicciones adicional que se proporcionó.

Tabla de Preprocessing

Preprocesamiento	Explicación	Función .py
Dummy Encoding	Codifica las columnas provistas con Dummy Encoding, evitando colinealidad.	aplicar_dummy_variables_encoding()
Hashing Trick	Codifica las comunas provistas con el Feature Hasher de sklearn, las cuales se transforman en un arreglo de n dimensiones, donde n se recibe como parámetro.	aplicar_hashing_trick()
Normalización (estándar o MinMax)	Normaliza las features recibiendo como parámetro un objeto entrenado Scaler de Sklearn, el cual debe ser entrenado previamente.	normalizar_datos()
Limpieza datos	Transforma missings del dataset a objeto NaN de numpy, y transforma feature presion_atmosferica_tarde a float, poniendo en nan los valores de string inválidos que tiene por defecto el dataset	limpiar_datos()
Reducción t-SNE(*)	Utilización de t-Stochastic neighbor embedding para reducir la dimensión del dataset.	reduccion_TSNE()
Reducción MDS (*)	Utilización de MDS para reducir dimensión del dataset	reduccion_MDS()
Reducción PCA	Utilización de Principal component analysis para reducir la dimensión del dataset	reduccion_PCA()
Eliminación de Features	Elimina las columnas que recibe por parámetro del dataframe provisto.	eliminar_features()
KNNImputer (*)	Imputa valores missing utilizando KNN.	imputar_missings_KNN()
IterativeImputer	Llena los missings de las features continuas mediante IterativeImputer de sklearn recibiendo dicho objeto como parámetro	imputar_missings_iterative()

Observaciónes:

- Los preprocesamientos (*) no se utilizan en ningún modelo por su elevado costo de tiempo, espacio, o simplemente por no ayudar al aprendizaje supervisado, si bien se probó usarlos.

- En algunos casos, las features que se deciden eliminar mediante eliminar_features() se toman de la decision de otros modelos, por ejemplo, en un procesamiento de arbol de clasificación se utiliza un ensamble random forest para decidir qué features son las más influyentes.

Tabla de modelos

Modelo	Preprocesamientos	AUC-ROC	Accuracy	Precision	Recall	F1 Score
Arbol decisión	Limpieza datos, Eliminación de Features, Dummy Encoding, IterativeImputer	0.85	0.84	0.71	0.49	0.58
KNN	Limpieza datos, Eliminación de Features, Dummy Encoding, Reducción PCA IterativeImputer	0.79	0.81	0.63	0.40	0.49
Naive bayes	Limpieza datos, Eliminación de Features, IterativeImputer	0.83	0.80	0.54	0.66	0.59
SVM	Limpieza datos, Eliminación de Features, Dummy Encoding, Normalización IterativeImputer	0.81	0.81	0.59	0.47	0.52
Random Forest	Limpieza datos Hashing Trick Eliminación de Features, IterativeImputer	0.88	0.85	0.76	0.47	0.58
Boosting	Limpieza datos, Dummy Encoding, Eliminación de Features, IterativeImputer	0.87	0.85	0.74	0.51	0.61
Red Neuronal	Limpieza datos, Eliminación de Features, Dummy Encoding, Normalización, IterativeImputer	0.71	0.85	0.75	0.47	0.57
Baseline		0.65	0.82	0.74	0.33	0.46

Conclusiones:

Comenzando por las conclusiones negativas, notamos en los modelos que tienen preprocesados más complejos como SVM y KNN una tendencia a dar métricas en general inferiores al promedio, dado que confiamos en los conocimientos que subyacen a las decisiones de qué preprocesamiento utilizar en cada caso, concluimos que la distribución del dataset no favorece a estos modelos los cuales además dependen de la distribución espacial del mismo. Como observación adicional, la Red Neuronal nos dió resultados mediocres en el dataset de holdout para la métrica AUC-ROC, a pesar de haber tenido buenas métricas con ella en el set de validación, y por lo tanto analizamos en su correspondiente notebook los posibles motivos de dicha observación.

Sin embargo, notamos que varios de nuestros otros modelos supervisados serían más satisfactorios para los propósitos de Flint.

Por un lado, creemos que los modelos más equilibrados fueron los ensambles de Boosting y Random Forest. Ya que si bien no tuvieron el mayor recall o F1 score, fueron quienes mantuvieron buenos valores para todas las métricas en general. Además son los modelos que alcanzan mayor score AUC-ROC, accuracy y precisión. Por lo tanto, si el cliente busca un modelo con buena performance general, serán estos los modelos que recomendaremos.

Sin embargo, Flint nos pide un análisis en para que modelo es mejor en caso de querer minimizar los falsos positivos o minimizar los falsos negativos, a lo cual corresponden los máximos puntajes de Precision y Recall respectivamente.

Para maximizar la Precision, el modelo que le recomendaremos al Flint, es Random Forest, que alcanza la accuracy y AUC-ROC score máximos y una Precision de 0.76, valor que también se alza como el mejor entre los otros modelos.

Por otro lado, si se quiere maximizar la Recall, entonces recomendaremos rotundamente nuestro modelo de Naive Bayes Gaussiano. El mismo, alcanzó un valor Recall de 0.66, muy por arriba de todos los demás modelos, y por lo tanto aquel con la mayor chance de atrapar las lluvias de hamburguesas positivas al día siguiente entre todos nuestros modelos.

Finalmente mediante modelos más complejos, logramos aumentar la performance de las predicciones en prácticamente todas las métricas en comparación a la baseline de la primera parte del trabajo práctico, la cual a pesar de ser extremadamente simple (una sola línea de código condicional), tuvo valores de métricas bastante bajos. Únicamente en la accuracy logró superar a otros modelos, siendo superior a Naive Bayes, KNN y SVM. Por lo tanto, en casi todas las situaciones, sería conveniente utilizar un modelo de machine learning más avanzado que la baseline, con la excepción de que se quiera trabajar con microcontroladores los cuales no dispongan de los recursos de los que disponen las computadoras convencionales.