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
Básico	Técnicas de FE comunes a todos los modelos. Tratamiento de missings, selección de features y OHE.	preprocesamiento_basico()
Básico GNB	Idéntico a preprocesamiento básico, pero eliminando también la columna 'llovieron_hamburguesas_hoy' pues Gaussian Naive Bayes trata features continuas.	
Normalización	Aplicación de StandardScaler de sklearn, para normalizar features continuas	normalizar_datos()
Reducción t-SNE	Utilización de t-Stochastic neighbor embedding para reducir la dimensión del dataset.	reduccion_TSNE()
Eliminación de Features	Invocado por Preprocesamiento 'Básico'. Elimina las columnas que recibe por parámetro del dataframe provisto.	eliminar_features()
Dummy Encoding	Invocado por Preprocesamiento 'Básico'. Codifica las columnas provistas con Dummy Encoding, evitando colinealidad.	aplicar_dummy_variables_encoding()
IterativeImputer	Invocado por Preprocesamiento 'Básico'. Llena los missings de las features continuas mediante IterativeImputer de sklearn.	imputar_missings_iterative()
Ordinal Encoding (*)	Codifica las columas provistas con Ordinal Encoding. Originalmente planeado para las columnas de dirección de viento (en orden radial).	aplicar_ordinal_encoding()
Reducción MDS (*)	Utilización de MDS para reducir dimensión del dataset	reduccion_MDS()
KNNImputer (*)	Imputa valores missing utilizando KNN.	imputar_missings_KNN()

Observación. 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.

Tabla de modelos

Modelo	Preprocesamientos	AUC-ROC	Accuracy	Precision	Recall	F1 Score
Arbol decisión	Básico	0.85	0.84	0.71	0.48	0.58
KNN	Básico, Reducción t-SNE	0.65	0.71	0.36	0.35	0.36
Naive bayes	Básico GNB	0.83	0.80	0.54	0.65	0.59
SVM	Básico, Normalización	0.84	0.85	0.78	0.46	0.58
Random Forest	Básico	0.88	0.85	0.766	0.4722	0.58
Boosting	Básico	0.87	0.85	0.75	0.51	0.60
Red Neuronal	Básico, Normalización	0.87	0.85	0.72	0.52	0.60
Baseline	Básico	0.65	0.82	0.74	0.33	0.46

Conclusiones:

Comenzando por las conclusiones negativas, es claro que nuestro modelo de KNN resultó en valores de métricas mucho menores a las que deseamos, por debajo de los demás modelos en todas ellas. Intuimos que esto se debe a la severa reducción de dimensionalidad que impusimos antes de predecir con ese modelo, lo cuál hicimos pensando en la tendencia de KNN a fallar para dimensiones grandes. Aún así, el notebook de KNN tenía un tiempo de ejecución tan elevado, que después de resultar mal en múltiples iteraciones optamos por tomar aquella con el mayor AUC-ROC score. Intentamos también cambiar la reducción t-SNE a MDS, pero dicha función pedía utilizar ~80GB de memoria ram, la cual no disponemos.

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 la Red Neuronal y el Ensamble de Boosting. Ya que si bien no tuvieron el mayor AUC ROC score, Precision, o Recall, sí fueron quienes mantuvieron buenos valores para todas estas métricas. 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 SVM, que mantiene una accuracy buena y una Precision de 0.78, valor que no alcanza ninguno de los demás 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 0.65, 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 y KNN. 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.