Tabla 1: pre-procesamientos.

Nombre de pre-procesamiento.	**Explicación**	**Nombre de la función.**
Convertir de numérico a categórico ordinal	Transforma una lista de features numéricos (float o int) a features categóricos ordinales, cuyas categorías son las que le pasamos a la función.	numerical_to_ordinal_categorical
One Hot Encoding	Aplica la técnica one hot (técnicamente dummy encoding) para encodear una variable categórica no ordinal	one_hot_encode
Hashing Trick	Aplica el hashing trick al feature pasado, agregando n_features columnas al dataframe. Ideal para features categóricos no ordinales de alta cardinalidad. Devuelve el df modificado	hashing_trick_encode
Eliminar features	Recibe una lista de features para dropearlos/eliminarlos del dataframe	drop_features
Escalamiento de features	Escala los features numéricos pasados en una lista usando StandardScalar (haciendo que los datos tengan media 0 y desvío estándar 1).	standard_scale
Recursive Feature Elimination	Selecciona como mínimo "min_features_to_select" features usando Recursive Feature Elimination, teniendo como estimador un árbol de decisión y usando cross validation de 3 folds. Elimina los features no seleccionados, y devuelve el X y los features eliminados.	select_features_RFECV
PCA	Aplica PCA sobre "features_to_reduce", reemplazándolos en el dataset por las “n_final_features” que explican la mayor varianza. Devuelve X y la varianza explicada.	reduce_dimension_of_features
Multivariate (Iterative) Imputer	Completa los missings del dataset (deben ser numéricos) usando Iterative Imputer (una implementación aproximada de MICE)	fill_with_iterative_imputer
Crear clase de NaNs	Para que en variables CATEGÓRICAS (pasadas en una lista "features_list"), tomemos a los NaNs como una categoría más.	nan_as_a_class
Univariate Imputer	Completa los NaNs de los features que se le pasa, usando alguna estrategia estadística como la median, mean o most_feaquent. Usa el SimpleImputer	fill_nan_with_simple_imputer
Convertir valores a NaN	Sirve para convertir valores no posibles/erráticos (que no corresponden a la naturaleza de dicho feature) y reemplazarlos por NaN	convert_values_to_nan
Convertir tipo de feature	Convierte los features pasados en la lista al tipo type, a fin de reducir el espacio en memoria que ocupa el dataset	convert_features
Obtener el mes de la fecha	Extrae el mes de un feature de tipo datetime64	extract_month_from_date

Lautaro Francetich, Agustín Gabrielli
Organización de Datos
FIUBA

Tabla 2: modelos.
Cabe destacar que la Precision, Recall y F1-Score fueron medidas usando un average “weighted”, es decir, que pesa los dos F1-Score (cada uno considerando a una clase como la positiva) y los promedia ponderando según los soportes de cada clase.

Modelo	**Preprocesamiento**	**AUC-ROC**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
1 - KNN	• Eliminar features • One Hot Encoding • Obtener el mes de la fecha • Multivariate (Iterative) Imputer • Escalamiento de features	0.85733	0.84433	0.83369	0.84433	0.82948
2 - LightGBM	• Eliminar features • Convertir tipo de feature • Obtener el mes de la fecha • Escalamiento de features • One Hot Encoding • Univariate Imputer • Recursive Feature Elimination	0.87685	0.85171	0.84226	0.85171	0.84135
3 - RandomForest	• Univariate Imputer • Eliminar features • Convertir valores a NaN • Convertir de numérico a categórico ordinal • One Hot Encoding • PCA	0.87456	0.85305	0.84427	0.85305	0.84024
4 - XGBoost	• Univariate Imputer • One Hot Encoding • Univariate Imputer • Escalamiento de features • Convertir valores a NaN • Eliminar features	0.87630	0.85143	0.84192	0.85143	0.84053
5 - Neural Network	• Eliminar features • Univariate Imputer • Convertir valores a NaN • One Hot Encoding • Multivariate (Iterative) Imputer • Escalamiento de features • Recursive Feature Elimination	0.88134	0.85385	0.84545	0.85385	0.84658

Lautaro Francetich, Agustín Gabrielli
Organización de Datos
FIUBA

Conclusión final

Luego de analizar, de entrenar, de testear y de probar a los modelos, podemos llegar a la conclusión de que el modelo que recomendamos por sobre los demás es NeuralNetwork, ya que tiene un AUC-ROC score superior a los otros modelos testeados. Esto quiere decir que aprendió a separar mejor las clases que el resto, a diferenciar si lloverá o no al día siguiente. Esto se ve además en el hecho de que dio mejor al resto en todas las métricas. Y una vez con el modelo entrenado, predecir no requiere de mucho tiempo y memoria (a diferencia de, por ejemplo, el KNN). También, el hecho de que le hayamos agregado regularización L2, dropout de neuronas, y early stopping hacen que generalice bien y por eso tenga buena performance en val-dev y test-holdout.

Por otro lado, claramente es muy superior a nuestro modelo baseline: éste último nos dio un accuracy del 0.82, mientras que la red neuronal nos dio 0.85. Pero además de esto, seguramente tenga un mucho mejor AUC-ROC (aunque no pudimos correrlo en el baseline).

Ahora, la desventaja de la red neuronal respecto al resto radica en que tanto el preprocesamiento (por el Iterative Imputer y el Recursive Feature Elimination sobre todo) como el entrenamiento llevan mucho tiempo (no tanto el predecir, como sí lleva de tiempo el KNN). Y ni hablar del GridSearch… Si se requiriese velocidad en el entrenamiento y en la predicción, tal vez LightGBM es uno de los mejores.

Si tuviésemos que elegir un modelo en base a tener la menor cantidad de falsos positivos, entonces debemos buscar aquél que tenga la mayor Precision (si digo que sí llueve mañana, que en verdad sea así); y esto lo cumple el NeuralNetwork.

Por otro lado, si quisiéramos tener una lista de todos los días que potencialmente lloverán hamburguesas al día siguiente sin preocuparnos demasiado si incluimos en la misma días que realmente no llovieron hamburguesas al día siguiente, estamos indirectamente diciendo que no nos importa si hay unos cuentos falsos positivos, pero queremos tener una alta exhaustividad, es decir, tener una alta cobertura de días que pueden llover. Eso es básicamente la definición de un alto Recall, que prioriza tener pocos falsos positivos (no queremos perdernos días en los que puede llover y no lo hayamos contemplado). El modelo que mejor dio en Recall fue, para variar, NeuralNetwork, seguido por poco de LightGBM.