FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

TP Integrador
2021

Análisis de Datos

Autor: Maximiliano Torti

Contenido

Ruta código Git

https://github.com/maxit1992/CEIA_AnD/tree/master/Clase8_TPIntegrador

Apartado 1

Se comenzó cargando el dataset con pandas y luego se utilizaron los métodos “head” y “describe” y las propiedades “shape” y “dtypes” para realizar un primer resumen.

De lo anterior se identificó que tenemos un dataset con 23 columnas (features) y alrededor de 145 mil registros. Se pueden consultar los significados de cada columna en Kaggle. Igualmente, los nombres de las columnas ya son muy explicativos. Se observó a priori que los features contienen diferentes tipos de datos y la presencia de valores faltantes.

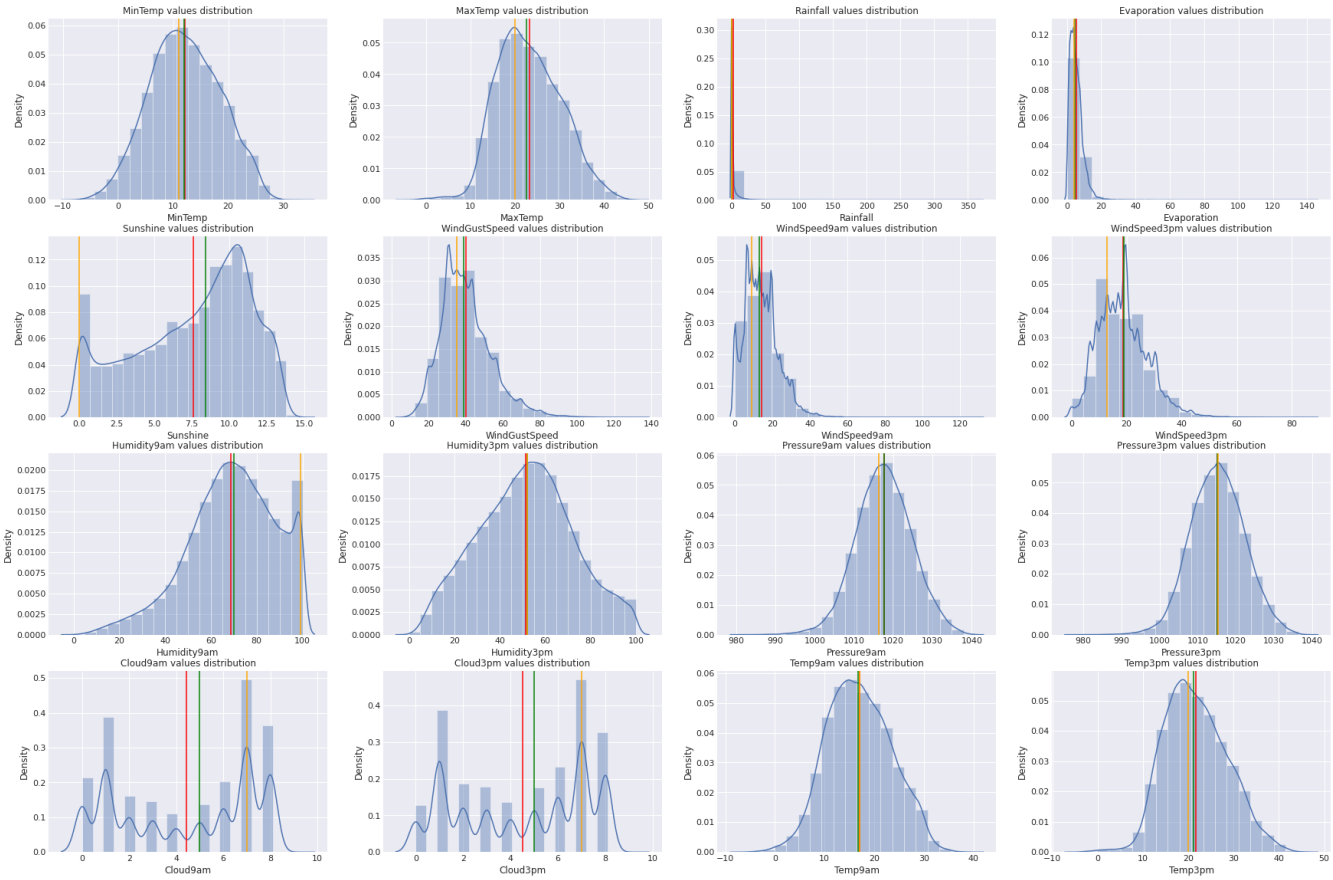
Por el tipo de problema planteado, se identificó como variable de salida ‘RainTomorrow’ y las 22 variables restantes serán las entradas de los modelos. También que el problema es de clasificación, donde la salida será 1 o 0 según se prediga que llueve mañana o no con datos del día de hoy.

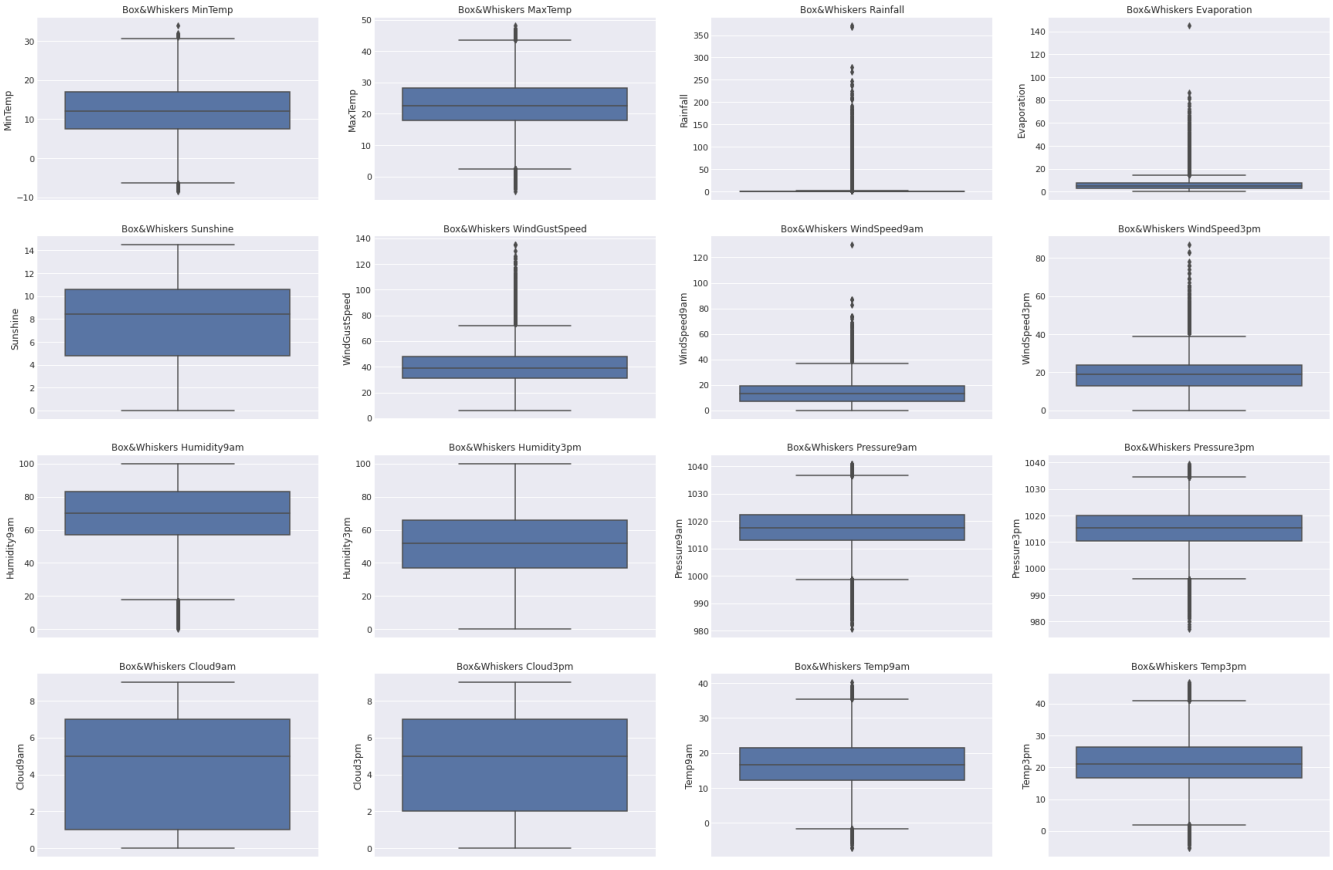
Posteriormente, se analizaron las variables de entrada según el tipo de dato:

- **Numéricas (16 en total)**

Utilizando el método “describe” y las funciones “distplot” y “boxplot” de la librería Seaborn se obtuvieron el dominio, valores máximos / mínimos, la distribución y el diagrama “Box-Whisker”. Adjuntamos tabla resumen y gráficas.

Variable	Valor	Min	Max	Distribución
MinTemp	Real	-8.5	33.9	Aproximadamente gaussiana. Sesgo leve hacia la derecha
MaxTemp	Real	-4.8	48.1	Aproximadamente gaussiana. Sesgo no tan leve hacia la derecha
Rainfall	Real	0	371	Irregular. Fuertemente puntiaguda en valores cercanos a 0. Con gran cantidad de valores extremos
Evaporation	Real	0	145	Puntiaguda en valores menores a 10. Con gran cantidad de valores extremos
Sunshine	Real	0	14.5	Irregular
WindGustSpeed	Real	6	135	Irregular
WindSpeed9am	Real	0	130	Irregular
WindSpeed3pm	Real	0	87	Irregular
Humidity9am	Real	0	100	Gaussiana recortada con pico adicional en 100
Humidity3pm	Real	0	100	Aproximadamente gaussiana
Pressure9am	Real	980.5	1041	Gaussiana
Pressure3pm	Real	977.1	1039	Gaussiana
Cloud9am	Entero	0	9	Aproximadamente gaussiana (a valores discretos) bimodal
Cloud3pm	Entero	0	9	Aproximadamente gaussiana (a valores discretos) bimodal
Temp9am	Real	-7.2	40.2	Gaussiana
Temp3pm	Real	-5.4	46.7	Gaussiana


MinTemp values distribution
MaxTemp values distribution
Rainfall values distribution
Evaporation values distribution
Sunshine values distribution
WindGustSpeed values distribution
WindSpeed9am values distribution
WindSpeed3pm values distribution
Humidity9am values distribution
Humidity3pm values distribution
Pressure9am values distribution
Pressure3pm values distribution
Cloud9am values distribution
Cloud3pm values distribution
Temp9am values distribution
Temp3pm values distribution
Density
MinTemp
MaxTemp
Rainfall
Evaporation
Sunshine
WindGustSpeed
WindSpeed9am
WindSpeed3pm
Humidity9am
Humidity3pm
Pressure9am
Pressure3pm
Cloud9am
Cloud3pm
Temp9am
Temp3pm


Box&Whiskers MinTemp
Box&Whiskers MaxTemp
Box&Whiskers Rainfall
Box&Whiskers Evaporation
Box&Whiskers Sunshine
Box&Whiskers WindGustSpeed
Box&Whiskers WindSpeed9am
Box&Whiskers WindSpeed3pm
Box&Whiskers Humidity9am
Box&Whiskers Humidity3pm
Box&Whiskers Pressure9am
Box&Whiskers Pressure3pm
Box&Whiskers Cloud9am
Box&Whiskers Cloud3pm
Box&Whiskers Temp9am
Box&Whiskers Temp3pm
MinTemp
MaxTemp
Rainfall
Evaporation
Sunshine
WindGustSpeed
WindSpeed9am
WindSpeed3pm
Humidity9am
Humidity3pm
Pressure9am
Pressure3pm
Cloud9am
Cloud3pm
Temp9am
Temp3pm

- **Categóricas (5 en total)**

Utilizando el método “describe” y “groupby” con “count” se obtuvieron el tipo de valor, la cardinalidad y la representación de cada categoría. La siguiente tabla resume los resultados.

Variable	Tipo	Informativa	Cardinalidad	Todas las categorías representadas	Posible codificación
Location	Nominal	Si	49	Si	- One Hot Encoding (no es recomendable debido a la alta cardinalidad). - Ordinal enconding o mean encoding
WindGustDir	Nominal	Si	16	Si	- One Hot Encoding - Codificar numéricamente según el cuadrante de dirección del viento.
WindDir9am	Nominal	Si	16	Si	
WindDir3pm	Nominal	Si	16	Si	
RainToday	Nominal	Si	2	Si	- Codificamos con 0 o 1 representando “No” o “Yes” respectivamente.

- **Otros**

La variable “date” es de tipo fecha, con formato YYYY-MM-DD. Con ese formato no es posible utilizarla para entrenar los modelos, es necesario codificarla. Se propone como método de codificación extraer únicamente el mes y utilizarlo como variable numérica. No se tomó el año ya que sería un feature que constantemente tomaría nuevos valores en producción. Otro método sería convertir la fecha a día del año.

- **Variable de salida**

La variable de salida se analizó mediante los métodos “describe” y “groupby / count”. Al ser un problema de clasificación binaria, es conveniente codificar la salida con 0 y 1 representando “No” y “Yes” respectivamente.

Variable	Tipo	Cardinalidad	Balance	Codificación
RainTomorrow	Categórica nominal	2	Desbalanceada con mayor cantidad de “No”	- Codificamos con 0 o 1 representando “No” o “Yes” respectivamente.

Apartado 2

Se utilizó la función “train_test_split” de la librería “sklearn” para particionar el dataset en 70% entrenamiento, 15% para validación de las técnicas y métodos que utilicemos y un 15% para un testeo final.

Apartado 3

A fin de conocer la proporción de valores faltantes para cada variable, se utilizó el método “isna().sum()” sobre el dataset de entrenamiento. A continuación, se muestran los valores obtenidos.

Date	*0 %*
Location	*0 %*
MinTemp	*1.03 %*
MaxTemp	*0.87 %*
Rainfall	*2.25 %*
Evaporation	*43.22 %*
Sunshine	*48.1 %*
WindGustDir	*7.13 %*
WindGustSpeed	*7.09 %*
WindDir9am	*7.26 %*
WindDir3pm	*2.94 %*
WindSpeed9am	*1.23 %*
WindSpeed3pm	*2.12 %*
Humidity9am	*1.83 %*
Humidity3pm	*3.08 %*
Pressure9am	*10.35 %*
Pressure3pm	*10.32 %*
Cloud9am	*38.3 %*
Cloud3pm	*40.81 %*
Temp9am	*1.23 %*
Temp3pm	*2.48 %*
RainToday	*2.24 %*
RainTomorrow	*2.26 %*

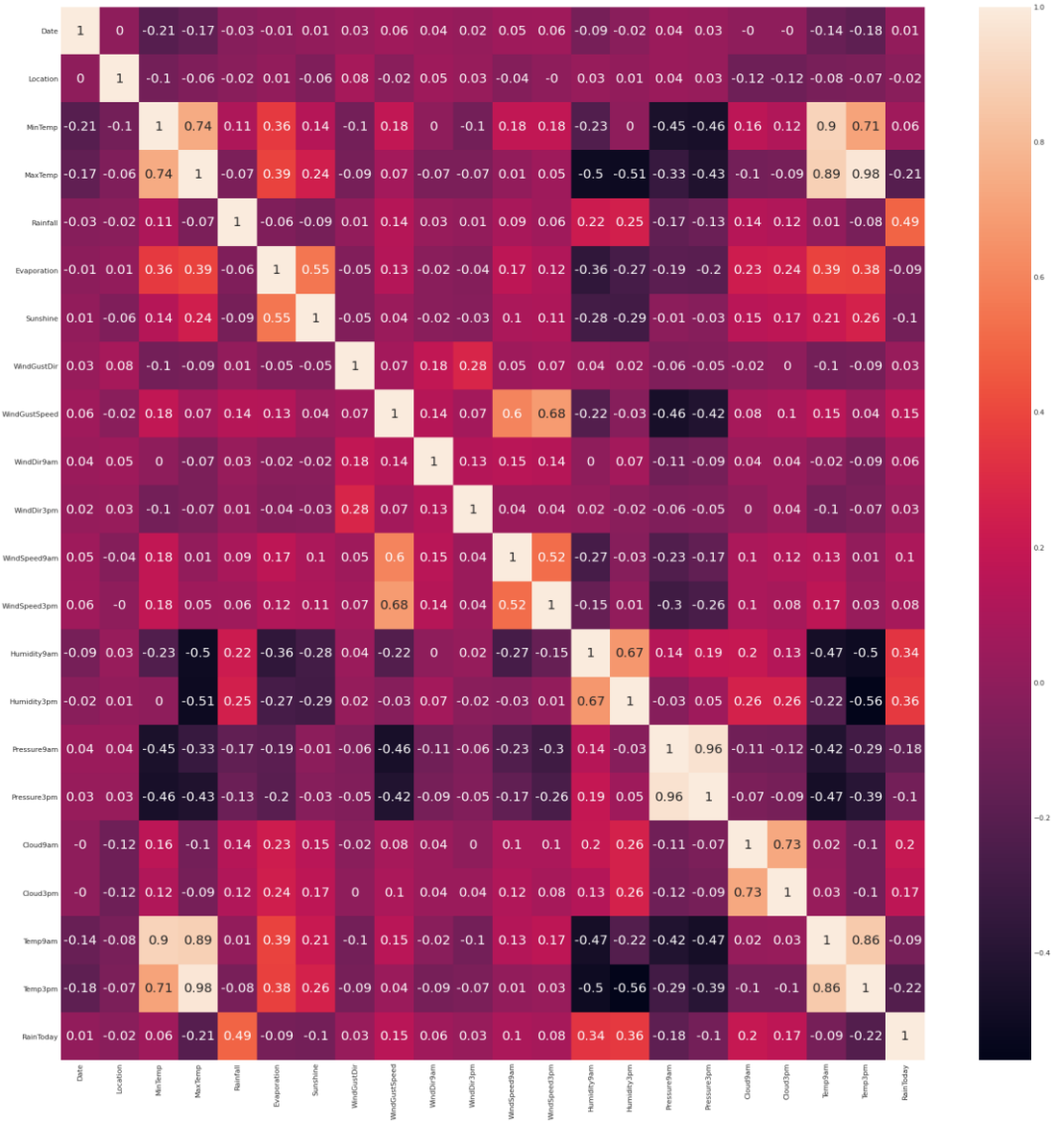
Posteriormente, con el método “head”, se inspeccionó cada variable para identificar los motivos de la ausencia de valores. También se utilizó el método “isna().sum()” combinando los “NaN” de diferentes variables en simultáneo. Las conclusiones que se obtuvieron fueron:

- RainTomorrow. El motivo de la falta de valores es desconocido (MCAR). Dado que es la salida en el problema “supervisado” planteado, se decide eliminar estos registros.
- Evaporation, Sunshine, Pressure9am, Pressure3pm, Cloud9am y Cloud3pm. La gran cantidad de datos faltantes se debe a que en algunas ciudades no se realizan mediciones de una o varias de estas variables (MNAR). Dada la gran proporción, se propone utilizar imputación por valor arbitrario fuera de la distribución o por valor de fin de cola.
- WindGustDir, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Temp9am, Temp3pm. En estos casos la mayor parte de valores faltantes se debe a que hay días al azar donde no se miden ciertas combinaciones como velocidad y dirección del viento, o temperatura y humedad (MAR). En su mayoría las proporciones de faltantes de estas variables es bajo, por lo que proponemos la imputación por mediana o moda para las numéricas y categoría faltante o frecuente para las categóricas.
- MinTemp, MaxTemp, WindGustDir9am, WindGustDir3pm, Rainfall, RainToday. El motivo de la falta de valores es totalmente desconocido y aleatorio (MCAR). Dado el bajo porcentaje de valores faltantes, se propone imputación con media, mediana o moda para las variables numéricas y categoría faltante o frecuente para las categóricas.

Nota: en un análisis mas elaborado se podría utilizar métodos multivariable como MICE para, por ejemplo, calcular el valor de Temp3pm a partir del resto de las temperaturas.

Respecto al tipo de codificación a utilizar en variables categóricas, se nombraron candidatos en tablas anteriores. La comparación de estas técnicas se analiza posteriormente al comparar los resultados de modelos.

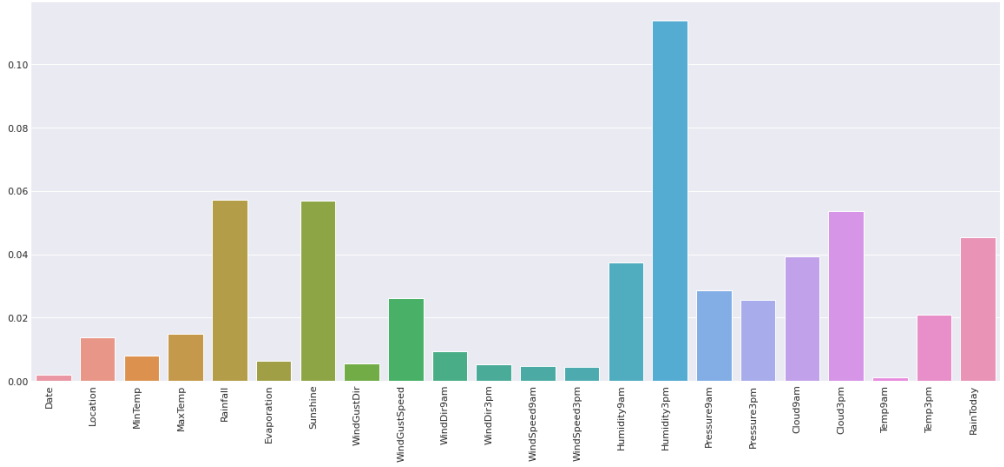
Previo al entrenamiento, se analizó la relación entre las entradas y la "importancia" de las mismas para la salida.
Para lo primero se utilizó el método "corr()" de Pandas que calcula la matriz de correlación con el coeficiente de Pearson (mide relaciones lineales). Para poder aplicar este método, dado que requiere variables numéricas, previamente codificamos aquellas variables de entrada que lo precisaban. Esta codificación nos puede afectar la relación, pero dado que la correlación la utilizaremos como un indicativo no es problema.



Se destaca:

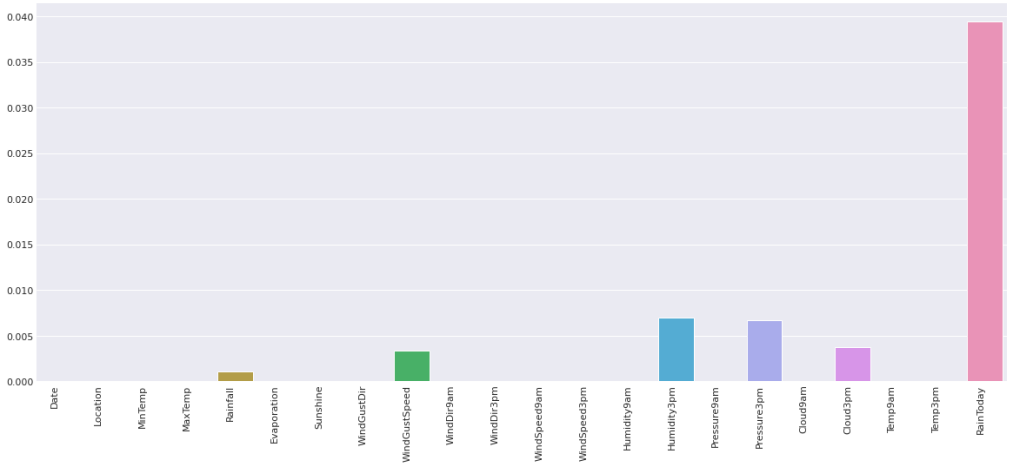
- Las temperaturas mínimas, máximas, 9am y 3pm están altamente correlacionadas.
- La velocidad máxima del viento esta levemente correlacionada con la velocidad a las 9am y 3pm.
- Las humedades a las 9am y 3pm están levemente correlacionadas.
- Las presiones 9am y 3pm están altamente correlacionadas.
- Las nubosidades a las 9am y 3pm están correlacionadas.

Para identificar las variables de entrada de mayor importancia se aplicó información mutua y método de Lasso. Al utilizar información mutua, se mide el coeficiente de “información mutua” entre las variables de entrada y la variable de salida y se selecciona aquellas variables de entrada con mayor “relación” con la salida. Ejecutando las funciones “select_features_mutual_info” y “barplot” se obtuvo el siguiente gráfico:



Con este estadístico, se destacan como variables de mayor interés Humidity3pm, Cloud3pm, RainToday, Rainfall y Sunshine.

Como segundo método de selección de variables se propone aplicar Lasso. Lasso es un método de regularización de regresión cuya característica es que anula las componentes de multiplicación (los “pesos”) de aquellos features que no son importantes en la regresión; es decir, selecciona las variables a considerar. Utilizando la clase “Lasso”, se obtuvo el siguiente resultado:



Aquí se destacan como variables de mayor importancia RainToday, Humidity3pm, Pressure3pm, Cloud3pm, WindGustSpeed y Rainfall. Los resultados entre ambos métodos son similares, pero asignan diferentes niveles de “importancia”.

Apartado 4 y 5

Al tratarse de un problema de clasificación binaria, se propone como modelos la Regresión Logística y Random Forest.
A partir del dataset de entrenamiento y las conclusiones del apartado anterior, se entrenaron ambos modelos para diferentes casos y luego se computó el AUC como medida de performance en los dataset de entrenamiento y validación.
En este apartado se utilizaron como ayuda los “Pipelines”, “Imputers” y “Encoders” de sklearn. En el caso de alguna operación que no se encuentre implementada (ejemplo convertir Date a mes), se realizó la tarea manualmente previa a la ejecución del pipeline.

Caso 1: caso completo, eliminando todos los NaN. Se utiliza el mes para codificar la fecha, ordinal encoding para las variables categóricas y 0/1 para ‘No’/’Yes’ en RainToday y RainTomorrow. Se comparó la performance entre escalado MinMax, escalado Standard y sin escalar.

	Train	valid
LogistiRegression no scale	0.881611	0.881279
LogistiRegression minmax scale	0.884779	0.884630
LogistiRegression standard scale	0.885483	0.884816
RandomForest no scale	1.000000	0.900085
RandomForest minmax scale	1.000000	0.900073
RandomForest standard scale	1.000000	0.899984

Observamos una buena performance en general de los modelos. Sin embargo, en este caso no podemos procesar el 60% de las muestras por no imputar los NaN. Random Forest tiende a hacer overfitting en el entrenamiento, pero generaliza bien igualmente. Regresión Logística muestra un buen equilibrio entre la performance de entrenamiento y validación, pero tiene peor performance

respecto a Random Forest como ya se intuía (por ser modelo lineal). El escalado mejora la performance de la regresión lo cual era esperable porque ayuda a la mejor convergencia del método, mientras que en Random Forest prácticamente no hay diferencia.

Caso 2: se imputa con mediana valores numéricos faltantes y con categoría "unknown" valores categóricos faltantes. Aplicamos solo escalado Standard.

	Train	Valid
LogistiRegression	0.862259	0.861019
RandomForest	1.000000	0.888134

La performance en general disminuyó, coherente con el hecho de haber imputado con mediana features con alta cantidad de NaN, lo cual distorsiona enormemente las distribuciones. El beneficio aquí es que podemos procesar todos los datos.

Caso 3: difiere del anterior que se imputa con mediana los valores numéricos faltantes en features con baja tasa de NaN (MinTemp, MaxTemp, RainFall, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Temp9am, Temp3pm) y se imputan con valor arbitrario (-1) los features con alta tasa de NaN (Evaporation, Sunshine, Pressure9am, Pressure3pm, Cloud9am, Cloud3pm).

	Train	valid
LogistiRegression	0.846967	0.845916
RandomForest	1.000000	0.888909

Para Regresión Logística hubo una baja apreciable de la performance. Esto se debe seguramente a la distorsión que genera la imputación por valor arbitrario en la distribución de algún feature importante para la regresión.

Caso 4: ídem anterior, pero se agrega columna de categoría faltante para features numéricos.

	Train	Valid
LogistiRegression	0.86326	0.863053
RandomForest	1.00000	0.889042

Se observa una mejora de ambos modelos, por lo que se concluye que la indicación de NaN es información importante para los modelos.

Caso 5: se utiliza One Hot Encoding para variables de baja cardinalidad.

	Train	valid
LogistiRegression	0.866419	0.866271
RandomForest	1.000000	0.887640

Mejora la performance de Regresión Logística, lo cual era esperable ya que One Hot Encoding preserva mejor la linealidad que asignar enteros al azar a las variables categóricas. Un punto interesante es que disminuye la performance de

Random Forest en validación. Se presupone que esto es por un aumento del overfitting.

Caso 6: se prueba con Mean Encoding en Location, teniendo en cuenta el promedio de la variable objetivo para cada ciudad. Los resultados siguen la dirección del caso anterior, donde mejora la performance de la regresión y empeora la de Random Forest.

Caso 7: partiendo del caso 5, se eliminan las variables de entrada correlacionadas, manteniendo aquella con mayor importancia para la salida.
- Se descarta Temp9am, MinTemp y MaxTemp y se mantiene Temp3pm.
- Se descartan variables de Humidity, Pressure y Cloud 9am y se mantienen las de 3pm.

	Train	valid
LogistiRegression	0.866206	0.86596
RandomForest	0.999922	0.88529

Se observa que prácticamente no se pierde performance en este caso por eliminar variables correlacionadas y con esto logramos disminuir la complejidad de los modelos y la carga de procesamiento.

Caso 8: partiendo del caso 5, se utilizan los features indicados como más importantes en al análisis con información mutua (Apartado 3).

	Train	valid
LogistiRegression	0.825632	0.826804
RandomForest	0.956898	0.792316

La performance en Regresión Logística disminuyó, pero en un valor relativamente razonable. La performance de Random Forest disminuyó mucho más de lo esperado, evidentemente fue muy optimista el límite fijado al eliminar variables.

Caso 9: partiendo del caso 5, se utilizan los features indicados como más importantes por el método de Lasso (Apartado 3).

	Train	valid
LogistiRegression	0.854641	0.854177
RandomForest	0.999039	0.836717

En este caso hay mejor performance que en el caso anterior. Se concluye que el límite tomado para el caso anterior fue muy optimista y también que los coeficientes de correlación no siempre indican lo que “el modelo necesita”.

Finalmente, utilizando el modelo del caso 5, se verifica la performance sobre el dataset de testeo, obteniendo un resultado acorde a lo esperado.

	test
LogistiRegression	0.868166
RandomForest	0.890142