TP INTEGRADOR: ANÁLISIS DE LLUVIA EN AUSTRALIA

Informe final

Descripción breve

Breve informe con los resultados de clasificación de días lluviosos y regresión de cantidad de lluvia.

Ing. Joel Spak

Joelspak45@gmail.com

Contenido

1. Carga de datos.

```
In [330]: df.head()
```

Out[330]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	Pressure3
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	22.0	1007.7	100
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	25.0	1010.6	100
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	30.0	1007.6	100
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	16.0	1017.6	101
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	33.0	1010.8	100

5 rows × 24 columns

2. Descripción de los datos

```
In [331]: df.describe()
```

Out[331]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3p
count	143928.000000	144159.000000	142152.000000	82658.000000	75616.000000	135159.000000	143645.000000	142351.000000	142759.000000	140907.0000
mean	12.195873	23.223176	2.361516	5.468616	7.611241	40.036564	14.044742	18.663164	68.877290	51.5374
std	6.398018	7.118770	8.479338	4.193871	3.785612	13.607212	8.915610	8.810276	19.029576	20.7967
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000	0.0000
25%	7.600000	17.900000	0.000000	2.600000	4.800000	31.000000	7.000000	13.000000	57.000000	37.0000
50%	12.000000	22.600000	0.000000	4.800000	8.400000	39.000000	13.000000	19.000000	70.000000	52.0000
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	24.000000	83.000000	66.0000
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000	100.0000

3. Análisis de la variable RainTomorrow

```
In [332]: print(pd.value_counts(df['RainTomorrow'], sort = True))

          ### Problema desbalanceado.
```

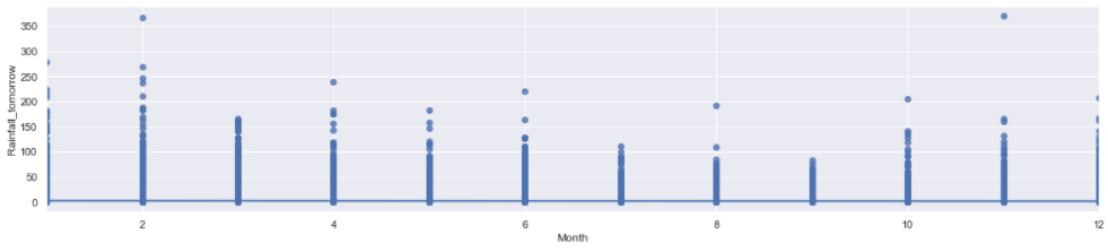
```
No     110281
Yes     31872
Name: RainTomorrow, dtype: int64
```

La variable de salida es **desbalanceada**: **en un 78% de los casos no llueve**. Esto se tendrá en cuenta a la hora de elegir un modelo para clasificar y para las métricas. Los valores "No" y "Yes" se traducen a ceros y unos para que los modelos puedan entrenar y predecir con variables numéricas.

4. Tratamiento de variables categóricas

4.1. Fecha

```
df["Month"]=pd.to_datetime(df["Date"]).dt.month
df=df.replace(to_replace="Yes", value=1)
df=df.replace(to_replace="No", value=0)
```



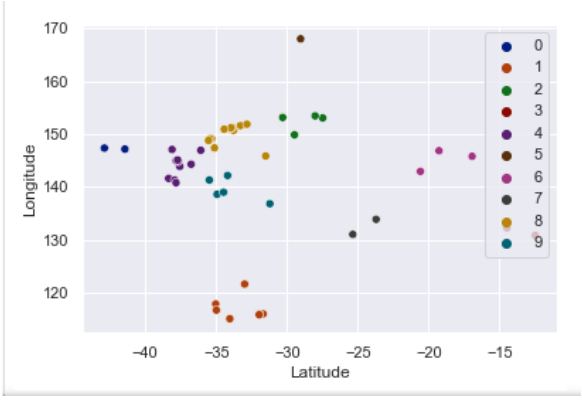
Rainfall_tomorrow vs Month

Para trabajar con la columna de Fecha, he graficado la cantidad de lluvia por mes. Donde veo que se puede clasificar por estación (menor lluvia en invierno, mayor en verano). Esto puede deberse al resto de las variables (condiciones de mayor humedad, por ejemplo), pero aun así lo tendremos en cuenta para el modelo. **No hay NaNs en las fechas.**

4.2. Dirección del viento

Decidí hacer una codificación binaria para las variables de dirección del viento, con las variables N, E, S, W. Por ejemplo, la dirección “NNE” se compone de “N” y “E”. Así, los NaNs se codifican como ceros (no son ni N, ni W, ni E, ni S).

4.3. Localidades



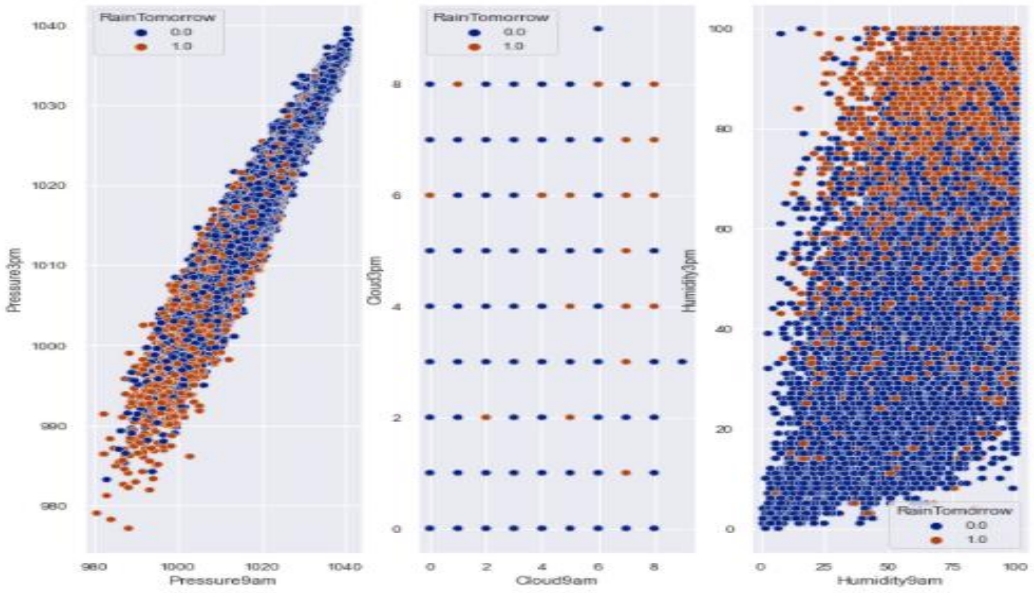
Puesto que hacer One Hot Encoding de las 49 localidades genera muchas columnas, decidí utilizar la librería Geopy para encontrar las longitudes y latitudes de cada ciudad y agruparlas en clusters (mediante K-means). La cantidad de clusters elegida fue 10. Se probó el modelo variando esta cantidad pero los resultados no se alteraban en gran medida.

Luego, se hace One Hot Encoding de los diez clusters (Generando diez columnas de datos con valores 1 o 0).

5. Distribuciones de las variables de entrada.

Se observan distribuciones aproximadamente gaussianas en la temperatura, presión, y además en la Humedad de las 3pm. En el resto de las variables no. Se ha trabajado con transformaciones para luego encontrar covarianzas.

Hay una fuerte correlación entre las temperaturas, y entre las presiones. El resto de las variables no tiene una correlación suficiente como para establecer una relación, que analizaremos con scatterplots.

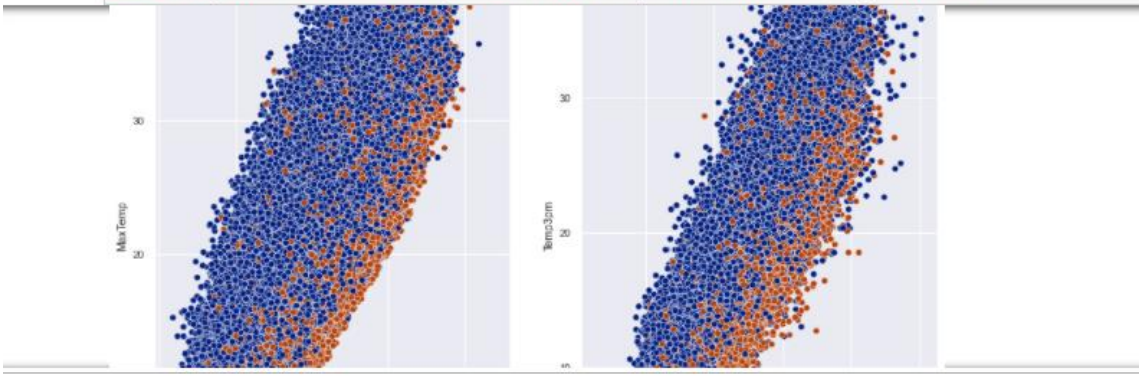


Presión 3pm vs Presión 9am, Cloud3pm vs Cloud9am y Humidity3pm vs Humidity9am

Observamos que en la presión hay una tendencia lineal, no así en el resto. Además, parece más probable que llueva mañana si hoy la presión es baja, ya sea en el horario 9am o 3pm (por eso, para imputar presiones tomaremos el promedio entre ambas y, en caso que alguna sea NaN, salteándola, de forma tal de ya hacer algunas imputaciones). La variable de nublado no parece tener influencia en la lluvia de mañana pero sí la Humedad 3pm, ya que mientras más alta, es más probable que RainTomorrow=1.

```
In [357]: fig,axes=plt.subplots(1,2,figsize=(13,6))
          sns.scatterplot(x="MinTemp", y="MaxTemp", data=X_train, ax=axes[0], hue="RainTomorrow", palette="dark")
          sns.scatterplot(x="Temp9am", y="Temp3pm", data=X_train, ax=axes[1], hue="RainTomorrow", palette="dark")

          #####3 ambos parecen casos linealmente separables. Parece ser importante la relación que hay entre ellas.
```



MaxTemp vs MinTemp y Temp3pm vs Temp9am.

En el caso de las temperaturas (la imagen puede verse mejor en el Notebook), hay una tendencia lineal. Un buen predictor para RainTomorrow parece ser una relación que hay entre MaxTemp y MinTemp, ya que el caso parece linealmente separable pero a través de una recta con pendiente (aproximadamente 45°). Usaremos como predictor la amplitud de temperatura (diferencia entre máxima y mínima).

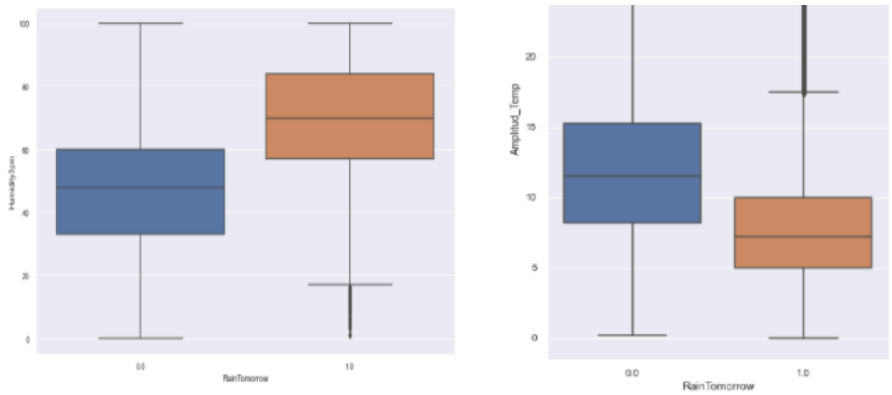


Diagrama de caja de Humedad3pm y Amplitud de temperatura.

6. Tratamiento de NaNs.

Las columnas Evaporation y Sunshine tienen una gran cantidad de datos faltantes (en algunos Clusters es del 100%). Por lo tanto la mejor solución es eliminarlas.

	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Cloud9am	Cloud3pm	Pressure9am	Pressure3pm	MaxTemp	MinTemp
4	3	1	0	0	2	2	27	28	2	2	1	1
8	2	13	2	3	2	4	49	51	24	24	0	1
2	2	4	3	3	0	0	35	34	2	2	0	0
5	1	1	0	0	0	0	2	1	0	0	0	0
6	1	1	0	0	0	0	30	32	0	0	0	0
9	0	0	0	0	0	0	38	45	0	0	0	0
1	2	16	0	4	1	5	47	50	15	15	0	0
0	0	0	0	0	0	0	55	55	18	18	0	0
7	1	2	0	0	0	0	33	32	0	0	0	0
3	0	1	0	0	1	17	0	15	0	0	0	1

Tabla “% de NaNs para cada cluster”

La tabla anterior muestra el porcentaje de NaNs que hay en cada cluster para cada columna. Debido a la gran cantidad de NaNs que tienen las variables Cloud9am y Cloud3pm, decidí eliminarlas.

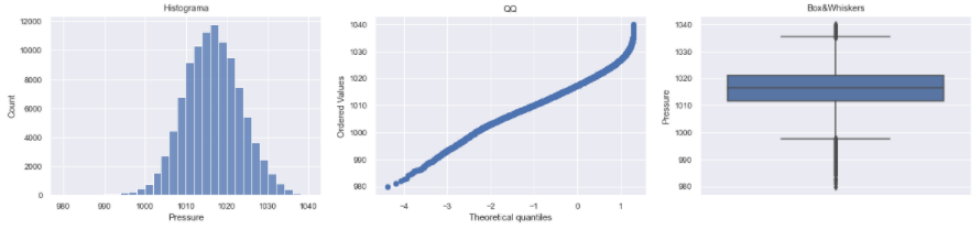
En el caso de las temperaturas, velocidades del viento y humedad de las 9am, la técnica aplicada para imputación es por media. En el caso de presión y humedad 3pm (los que hasta ahora considero mejores predictores), decidí imputarlos por KNN (usando también la amplitud de temperatura para ello).

7. Outliers

```
In [369]: ########3 OUTLIERS
upper,lower=outlier_diagnostic_plots(X_train, "Pressure", 1.5)
print(upper, lower)
outliers = np.where(X_train['Pressure'] > upper, True,
                    np.where(X_train['Pressure'] < lower, True, False))
xtrain_trimmed = X_train.loc[~outliers, ]
X_train.shape, xtrain_trimmed.shape

1035.3249999999998 997.5250000000001

Out[369]: ((112609, 37), (111796, 37))
```



Outliers de presión

Para el caso de los outliers, realicé un análisis para todas las variables, donde alrededor de 2000 datos en promedio (de los 112000 del set de entrenamiento) son outliers. Puesto que considero que es una baja cantidad, opté por hacer un tratamiento con RobustScaler de sklearn antes de ingresar las variables a los modelos de entrenamiento. RobustScaler, a diferencia de StandardScaler o MinMaxScaler, permite elegir un rango intercuartil para escalar los datos dentro de ese rango. He elegido un rango (5,95) puesto que es el que mejores resultados me dio.

8. Selección de Features

8.1. Información mutua

Rainfall	5.596038
WindGustSpeed	2.419181
WindSpeed9am	0.662279
WindSpeed3pm	0.395122
Humidity9am	3.840100
Humidity3pm	11.646023
RainToday	4.153677
Verano	0.000000
Otoño	0.000000
Invierno	0.369592
Primavera	0.000000

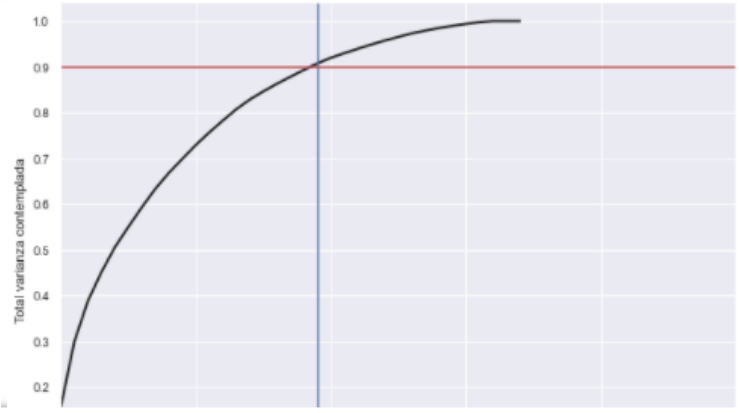
[0]	0.000000
[1]	0.010005
[2]	0.092788
[3]	0.000000
[4]	0.684566
[5]	0.000000
[6]	0.000000
[7]	0.151822
[8]	0.000000
[9]	0.145442
Amplitud_Temp	7.111507
Pressure	2.964238

WindGustDir_N	0.373574
WindGustDir_S	0.381228
WindGustDir_W	0.000000
WindGustDir_E	0.078412
WindDir9am_N	0.072710
WindDir9am_S	0.000000
WindDir9am_W	0.308092
WindDir9am_E	0.520670
WindDir3pm_N	0.271745
WindDir3pm_S	0.017042
WindDir3pm_W	0.000000
WindDir3pm_E	0.392809

Se muestra 100 * MI para tener números que puedan apreciarse mejor. Sigue que son buenos predictores Rainfall, Humidity 9am, Humidity3m, RainToday, Amplitud_Temp y presión, no así muchas direcciones del viento o estaciones del año.

8.2. PCA

Aplicando el algoritmo de PCA, la cantidad de columnas 35 se reduce a 20.



9. Entrenamiento de Modelos

9.1. Métricas

Al ser la variable de salida no balanceada, es importante definir las métricas con las cuales voy a comparar los modelos empleados. Para este tipo de clases, "recall" y "precisión" son las variables más importantes puesto que reflejan mejor la predicción de los días de lluvia, que es la clase minoritaria. Para englobarlas, usé F1_score, que es la media armónica entre Recall y Precision ($\frac{2PR}{P+R}$). Se le puede otorgar mayor peso a alguna de ellas, pero en este caso las consideré de mismo peso.

9.2. KMeans

Sólo a modo de prueba puesto que al ser no supervisado, no se esperan buenos resultados. Se entrena con RobustScaler y PCA.

9.3. Regresión logística con SGD

Se implementa el código dado en clase. Los resultados son significativamente mejores que con KMeans.

9.4. Regresión logística con SKLearn

Se prueban diferentes hiperparámetros. En particular, permite adoptar class_balanced=[None, 'balanced'] para tratar con clases desbalanceadas. He corrido con penalty=L2, y los valores de regularización inversa C y máxima cantidad de iteraciones las he encontrado **utilizando K-Folds.**

He probado además no implementar PCA y continuar con todas las columnas de mi entrenamiento, obteniendo mejores resultados.

9.5. Random Forest Classifier y AdaBoost Clasiffier

En orden a explorar nuevos modelos de clasificación, implementé árboles que hemos dado en Análisis de datos para hacer distintas comparaciones. La gran mayoría de los hiperparámetros fueron seteados en automático excepto por class_balanced de RFC que probé poner en 'balanced'.

10. Resultados clasificación

	Modelo	RobustScaler	PCA	Accuracy	Precision	Recall	F1
0	KMeans	Si	Si	0.526015	0.245368	0.556130	0.340503
1	Log. Reg. SGD	Si	Si	0.825965	0.709385	0.354062	0.472363
2	Log. Reg.	Si	Si	0.829874	0.693282	0.406719	0.512674
3	Log. Reg.	Si	No	0.845120	0.722614	0.480536	0.577222
4	Log. Reg. Balanced	Si	No	0.782607	0.504003	0.752382	0.603642
5	Log. Reg. Balanced K-Folds	Si	No	0.782643	0.504057	0.752544	0.603732
6	RFC	Si	No	0.856706	0.764130	0.504442	0.607706
7	RFC Balanced	Si	No	0.855640	0.776135	0.483282	0.595660
8	AdaBoost Classifier	Si	No	0.844836	0.721858	0.479567	0.576281

El mejor modelo, de los vistos en clase, es la regresión logística balanceada con regularización obtenida mediante K-Folds, que da un F1=0.6037.

Es de destacar que, sin un análisis intenso, el desempeño de RFC es aún mejor que el mencionado anteriormente y motiva a seguir estudiando e incorporando distintos tipos de modelos de IA.

11. Regresiones

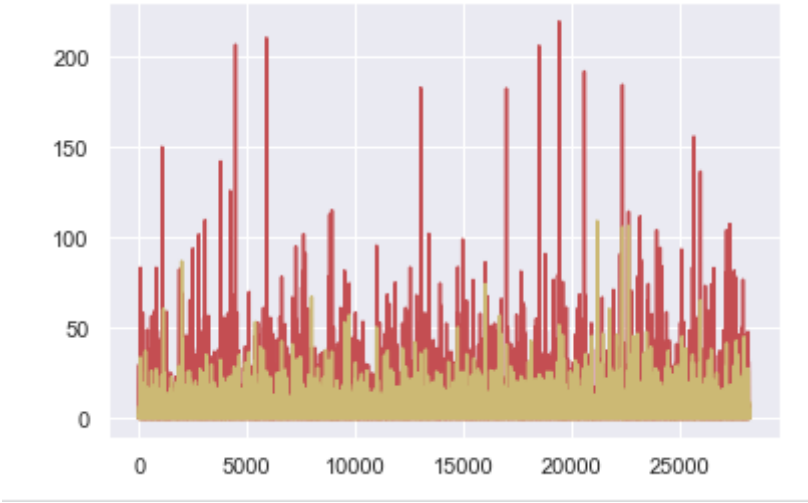
Para el caso de las regresiones, decidí no usar las estaciones del año puesto que Información Mutua daba valores muy bajos. Empleé el modelo de regresión lineal clásico al cual también le añadí polinomios de orden 2 y 3. También probé con regresiones Ridge y Lasso, variando alpha entre 0.1 y 10.

Las métricas que tomé son MSE y R2_score.

	Reg. Lineal	Ridge alpha=0.1	Ridge alpha=0.5	Ridge alpha=1	Ridge alpha=5	Ridge alpha=10	Lasso alpha=0.1	Lasso alpha=0.5	Lasso alpha=1	Lasso alpha=5	Lasso alpha=10
MSE_train	55.518389	55.979807	57.595925	58.971158	63.883148	66.062962	55.979807	57.595925	58.971158	63.883148	66.062962
R2_train	0.206824	0.200232	0.177143	0.157496	0.087320	0.056177	0.200232	0.177143	0.157496	0.087320	0.056177
MSE_test	53.173657	53.666840	55.318186	56.683474	61.488573	63.616057	53.666840	55.318186	56.683474	61.488573	63.616057
R2_test	0.211769	0.204458	0.179979	0.159740	0.088511	0.056974	0.204458	0.179979	0.159740	0.088511	0.056974

Reg. sólo térm cuad.	Reg. Polin. 2°	Reg. Polin. 3°
55.239076	47.637770	39.688335
0.210815	0.319413	0.432984
51.997442	46.638786	52.822854
0.229205	0.308640	0.216969

La métrica R2 es baja en todos los casos, pero el mejor modelo **es la regresión polinómica de orden 2**, ya que la de orden 3, tiene mejores resultados en el entrenamiento pero falla a la hora de generalizar. Es, además, la de menor MSE.



Regresión polinómica de orden 2. En rojo la cantidad de lluvia real, en amarillo la predicción.