Informe Trabajo Integrador

Análisis de datos

CEIA 2021

Contenido

Introducción:

A partir de la propuesta de trabajo final para la materia de Análisis de Datos, se elabora el presente informe final con el objetivo de detallar los métodos utilizados para la resolución del trabajo integrador.

El dataset brindado para realizar este trabajo proviene de *kaggle* y el objetivo es predecir la variable de salida *RainTomorrow*. El dataset contiene aproximadamente mas de 10 años de observaciones meteorológicas de varios lugares de Australia

El desafío consiste en realizar un análisis exploratorio inicial utilizando diferentes técnicas para el análisis de datos. Luego, se realizará la preparación de los datos que serán el input del modelo de machine learning, el entrenamiento, evaluación e interpretación de resultados de los diferentes modelos. Finalmente se deberán desarrollar las conclusiones.

Análisis exploratorio inicial

Visualización de las primeras filas del dataset

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	WNW	20.0	24.0	71.0
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	WSW	4.0	22.0	44.0
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	WSW	19.0	26.0	38.0
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	E	11.0	9.0	45.0
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	NW	7.0	20.0	82.0

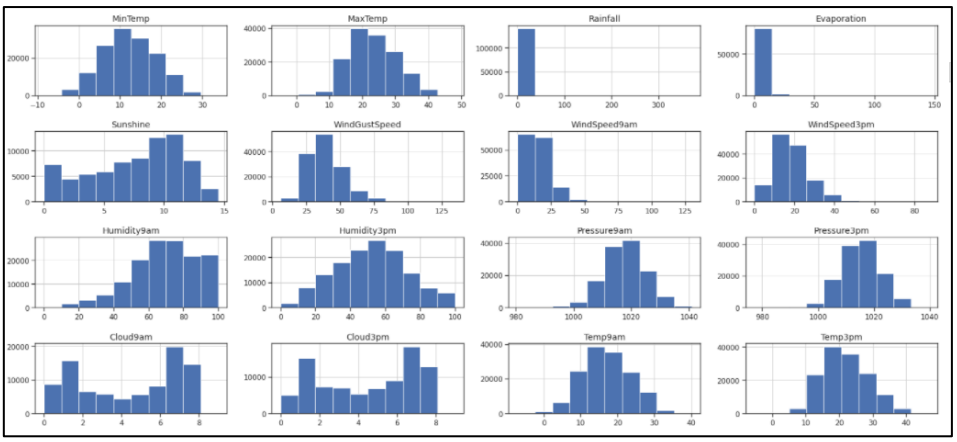
El dataset contiene 145460 filas y 24 columnas.

Identificación de tipos de datos:

Las variables categóricas en primera instancia son *Date, Location, WindGustDir, WindDir9am, WindDir3pm, RainToday*. La variable a predecir es *RainTomorrow* (variable de salida) y las demás variables son numéricas. Todas las variables menos la variable a predecir (*RainTomorrow*) son input para el modelo (variables de entrada).

Análisis por tipo de variable: numérica

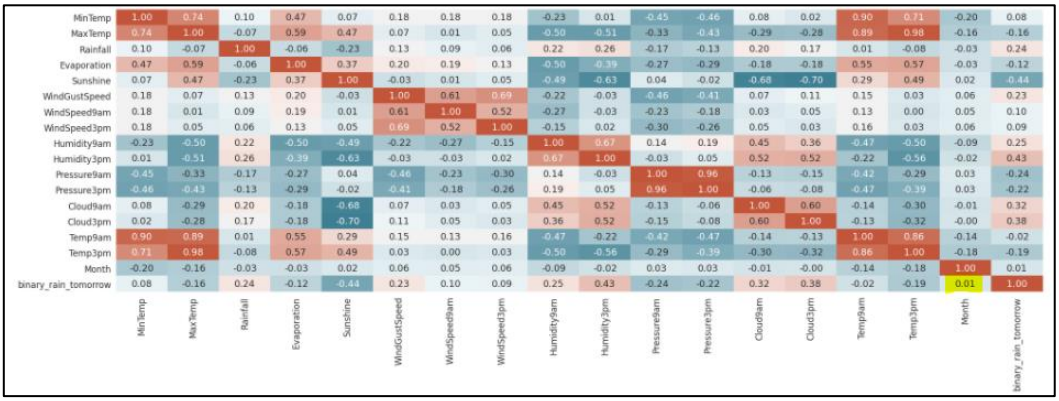
Se observa la distribución de los datos a partir de la creación de histograma. La mayoría de las variables muestra seguir una distribución normal.



Análisis por tipo de variable: categórica

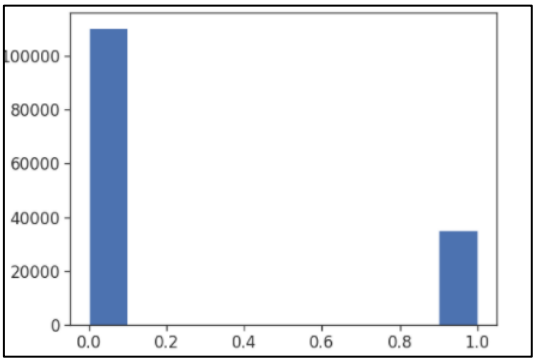
Al realizar el análisis de cardinalidad de las variables categóricas se llega a la conclusión de que Date y Location tienen alta cardinalidad.

Inicialmente durante el análisis se identificó a la variable Date de importancia para el modelo ya que el mes podría estar relacionado con la variable de salida. Más adelante, durante el proceso se desestimó ya que al realizar un análisis de correlación entre el mes extraído de la columna Date y la variable de salida, se pudo concluir que no estaban correlacionadas.



Análisis de la variable de salida y balance de clases:

Se realizo el análisis del balance de clases en la variable de salida aplicando una función lambda donde “No” = 0 y “Si” =1 y se identificó que las clases estaban desbalanceadas, como se muestra en la imagen:



Preprocesamiento del dataset

Ejemplo Modelo 1

Se realiza una conversión binaria como técnica de codificación de la variable de salida para poder utilizarla en el entrenamiento y se crea la nueva variable de salida: “`binary_rain_tomorrow`”

Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow	binary_rain_tomorrow
1007.7	1007.1	8.0	NaN	16.9	21.8	No	No	0
1010.6	1007.8	NaN	NaN	17.2	24.3	No	No	0
1007.6	1008.7	NaN	2.0	21.0	23.2	No	No	0
1017.6	1012.8	NaN	NaN	18.1	26.5	No	No	0
1010.8	1006.0	7.0	8.0	17.8	29.7	No	No	0

Manejo de datos imbalanceados

Como se observó en el análisis exploratorio inicial, los datos están fuertemente desbalanceados en favor de los casos de que no llueva al día siguiente. Frente a esto se pueden adoptar distintas estrategias:

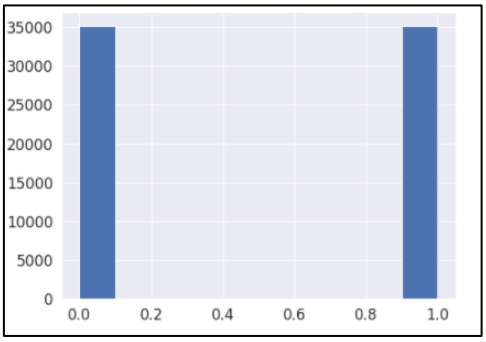
Upsampling: aumentar la cantidad de muestras de la clase minoritaria (agregando muestras reales o artificiales).

Downsampling: reducir la cantidad de muestras de la clase mayoritaria.

Seleccionar cuidadosamente la métrica de evaluación: no hacer nada en la preparación del dataset, y elegir una métrica de performance adecuada para este escenario (TPR/TNR/AUC/Precision/Recall/etc.)

Para continuar con este análisis trabajando con datos originales se procederá con la segunda opción.

Se genera un nuevo dataset que contiene clases balanceadas y se verifica que realmente quedaron balanceadas:



El nuevo dataset "`df_downsampled`" contiene 70288 filas y 24 columnas.

Tratamiento de NaNs

Se identifica un gran porcentaje de NaNs:

```
Cantidad de filas con nans (89040, 23)
Cantidad de filas sin nans (56420, 23)
```

Se realiza un análisis de la cantidad de NaNs del nuevo dataset y el tratamiento de los mismos.

Para el tratamiento de NaNs se utiliza la **imputación estadística** con la cual se completaron las variables categóricas con la moda y las variables numéricas con la media.

Se verifica que las distribuciones no hayan cambiado respecto al dataset original y que ya no haya filas con NaNs:

```
Cantidad de filas con nans (0, 24)
Cantidad de filas sin nans (70288, 24)
```

Ingeniería de features básica

- Para la variable *RainToday* se identifica que solo tiene dos clases, por lo cual se realiza una transformación con one hot encoding, generando dos nuevas columnas: `raintoday_No` y `raintoday_Yes`.

- Para las demás variables categóricas se hizo el mismo tratamiento, pero al analizar la correlación con la variable de salida, éstas no eran significativas, por lo cual se decidió omitirlas y excluirlas del dataset.

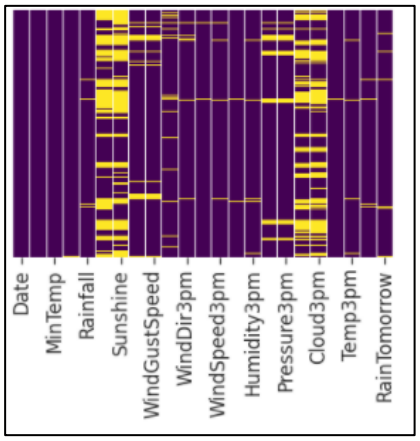
Ejemplo Modelo 2

Tratamiento de outliers

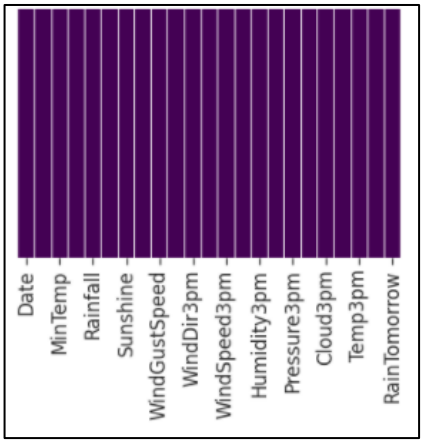
Se crearon funciones para encontrar los outliers dentro del dataset y se identificaron los limites superior e inferior de las variables: Rainfall, Evaporation, WindGustSpeed, WindSpeed9am y WindSpeed3pm. Se eliminaron los valores atípicos para que los mismos no afecten a los modelos de clasificación.

Tratamiento de NaNs

Se identifico un gran porcentaje de NaNs, y para tratarlos se utilizó el método de imputación estadística, completando los nans de las variables categóricas con la moda y los de las variables numéricas con la mediana.



Antes de aplicar método de imputaciónestadística



Después de aplicar imputación estadística

Ingenieria de features básica

Se utiliza el método de one hot encoding para todas las variables categóricas, menos para la variable Date, la cual se excluye del análisis.

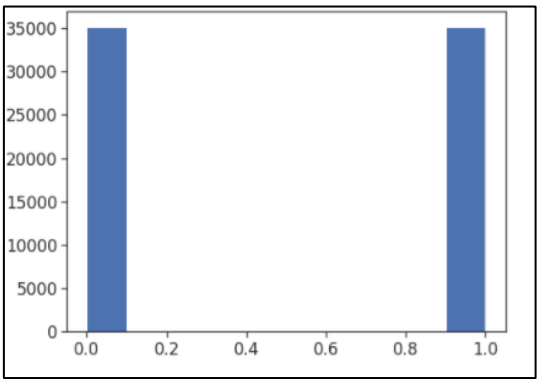
Para las variables numéricas, se realiza la normalización para la aplicación de los diferentes modelos de clasificación.

Ejemplo Modelo 3

Para el preprocesamiento de datos, se crea la variable `binary_rain_tomorrow` utilizando la función lambda.

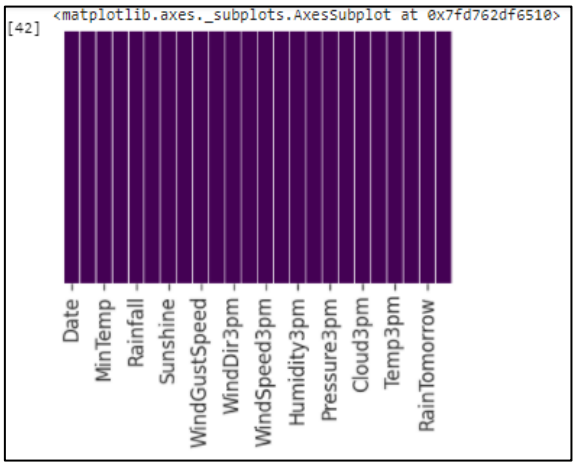
Manejo de datos imbalanceados

Se aplica la técnica de downsampling y se genera un nuevo dataset “df_downsampled” para el balance de las clases.



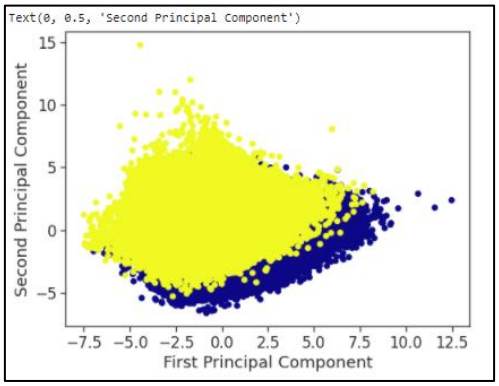
Tratamiento de NaNs

Se completan los datos categóricos con la moda y los datos numéricos con la media, obteniendo un dataset final sin nans:



Ingeniería de features básica

- Se elimina la variable Date para seguir con el desarrollo, ya que en el primer modelo implementado se definió que no aporta valor al análisis.
- Se realiza la transformación de las variables categoricas utilizando one hot encoding, lo cual genera una gran cantidad de columnas adicionales.
- Se realiza la normalización de las variables numericas con `StandardScaler()`
- Se utiliza PCA como método de reducción de dimensionalidad:



Implementación de modelos de ML

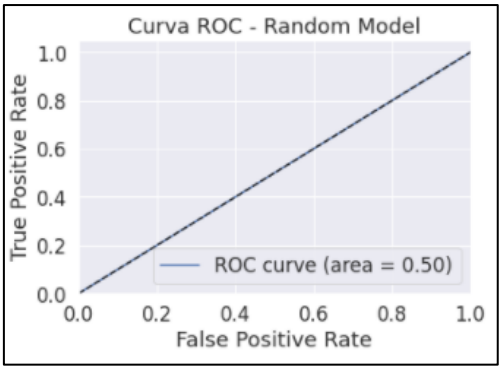
Como métrica de evaluación de los modelos se utilizó AUC porque mide la capacidad del modelo para predecir una mayor puntuación para ejemplos positivos en comparación con ejemplos negativos.

Ejemplo Modelo 1

Modelo Base

Se crea un modelo base a partir de una clase con fit y transform que devuelve resultados completamente random. El mismo se utilizará como instrumento de comparación de los siguientes modelos desarrollados.

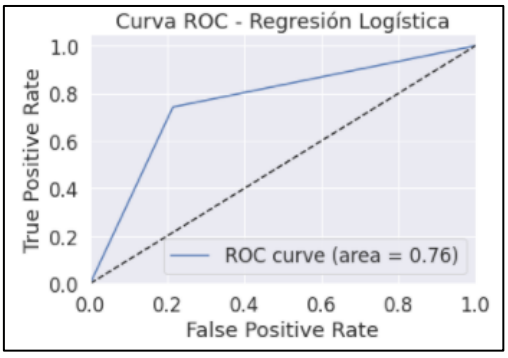
AUC: 0.5



Regresión Logística

El segundo modelo implementado es la regresión logística, ya que este método trata de modelar la probabilidad de una variable cualitativa binaria (dos posibles valores), en este caso "llueve mañana" o "no llueve mañana", en función de una o mas variables independientes. Este método se debe aplicar con datos normalizados, quiere decir que los valores de las variables de entrada tienen que oscilar entre 0 y 1. Es por eso que en un paso anterior se utiliza `MinMaxScaler()`

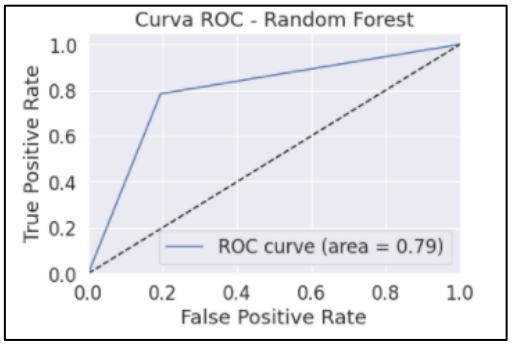
AUC: 0.76



Random Forest

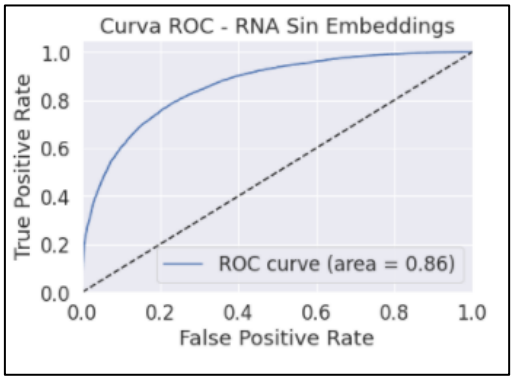
El tercer modelo implementado es el de *random forest*, un algoritmo de clasificación conformado por muchos árboles de decisión. Utiliza el método de ensamble y la aleatoriedad de las características al construir cada árbol para crear el *random forest* de árboles no

correlacionados cuya predicción conjunta sea más precisa que la de cualquier árbol individual. Se obtuvo un AUC de 0.79



Deeplearning

El ultimo modelo utilizado es el de *deeplearning*. Se utilizo un solo *hidden layer* y datos normalizados. Como función de activación en la salida se utilizó *sigmoid* ya que, al realizar un análisis de la variable de salida, se sabia que la misma iba a ser 0 o 1.



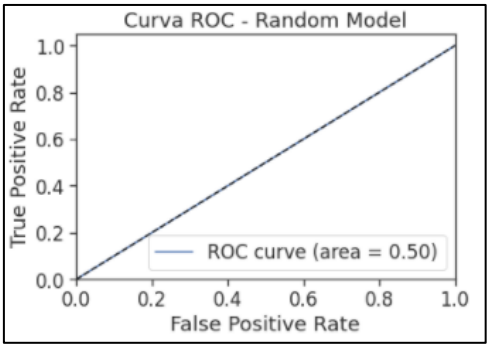
El modelo de DL (con un AUC del 86.16%) tiene un mejor área bajo la curva, no presenta baches por lo cual es un indicador de que clasifica bien (es una curva suave).

Ejemplo Modelo 2

Modelo Base

Se crea un modelo base a partir de una clase con fit y transform que devuelve resultados completamente random. El mismo se utilizará como instrumento de comparación de los siguientes modelos desarrollados.

AUC: 0.5



Regresión logística

El segundo modelo implementado es la regresión logística. Este método se debe aplicar con datos normalizados, quiere decir que los valores de las variables de entrada tienen que oscilar entre 0 y 1. Es por eso que en un paso anterior se utiliza `MinMaxScaler()`

Accuracy: 85 %

Regresion logistica

```
[67] from sklearn.metrics import confusion_matrix
     from sklearn.metrics import accuracy_score
     model = LogisticRegression(max_iter=500)
     model.fit(X_train, y_train)
     predicted=model.predict(X_test)

     conf = confusion_matrix(y_test, predicted)
     print ("The accuracy of Logistic Regression is : ", accuracy_score(y_test, predicted)*100, "%")
```

```
The accuracy of Logistic Regression is :  84.91681561941428 %
```

XGBoost

El tercer modelo implementado es XGBoost, el cual arrojó mejores resultados que la regresión logística:

```
Accuracy = 0.8614510289197489
ROC score = 0.8930273284320036
              precision    recall  f1-score   support

           0       0.88      0.95      0.91     33987
           1       0.76      0.55      0.64      9651

    accuracy                           0.86     43638
   macro avg       0.82      0.75      0.78     43638
weighted avg       0.85      0.86      0.85     43638
```

Ejemplo Modelo 3

XGBoost

Se implemento el modelo XGBoost el cual arrojo un accuracy de 79%:

```
import xgboost as xgb
xgb = xgb.XGBClassifier()
xgb.fit(X_train, y_train)
y_pred = xgb.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE: %f" % (rmse))
```

```
RMSE: 0.464908
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: %.2f" % (accuracy * 100.0))
```

```
Accuracy: 78.39
```

Referencias

Código fuente en git: https://github.com/paomar22/IA/tree/master/ADD

Ejemplo Modelo 1: ADD- Trabajo Integrador- Modelo 1.ipynb

Ejemplo Modelo 2: ADD- Trabajo Integrador- Modelo 2.ipynb

Ejemplo Modelo 3: ADD- Trabajo Integrador- Modelo 3.ipynb