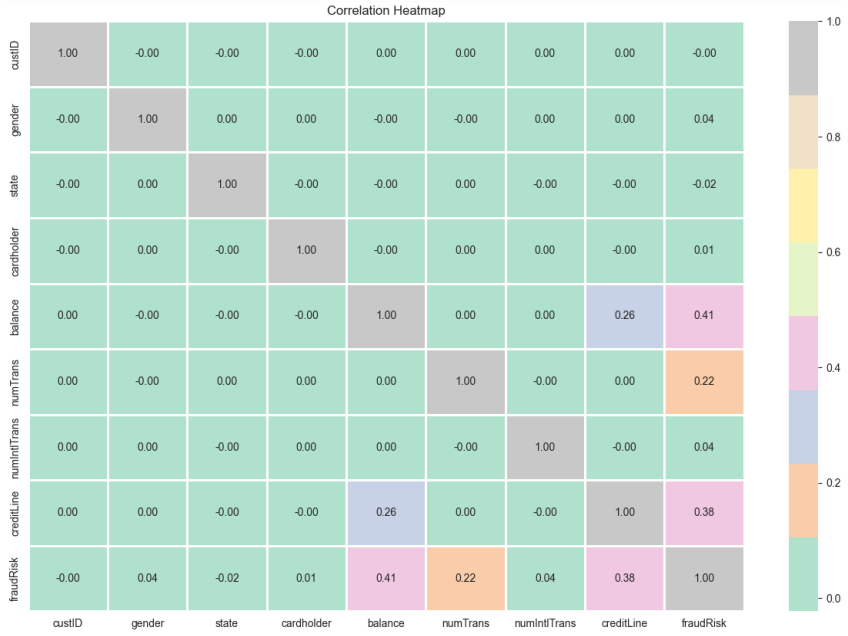
Datos, ciberseguridad e inteligencia artificial
Fecha: 26-06-2025.
Alumno: Taisen Romero Bañuelos.

Ejemplo para detección de fraude

El dilema de los datasets grandes son los tiempos de espera de entrenamiento de los modelos. En mi propuesta inicialmente quise hacer un modelo de Random Forest y otro de SVM, sin embargo, los detalles sobre Python, el refinamiento de algunas cosas del modelo y su proceso y los tiempos de espera me hicieron decidir que lo mejor sería centrarme en un modelo. Elegí centrarme en SVM porque es el modelo con el que empecé a trabajar y también porque el paquete con el que se aplica SVM ofrecía un parámetro para tratar el desbalance de clases, entonces, este sencillo argumento compensa mi inexperiencia en Python y el desbalance de clases.
Realmente invito a leer mi notebook ya que ahí hablo más en profundidad de todo, pero a manera de resumen me gustaría destacar que probé algunos modelos de SVM, y debido al rendimiento que ha tenido cada uno he persistido en refinar cada vez más el proceso de entrenamiento. Personalmente creo que la detección de falsos positivos y falsos negativos es complicada debido a no sólo el desbalance de clases, sino también a la baja correlación que hay entre variables.

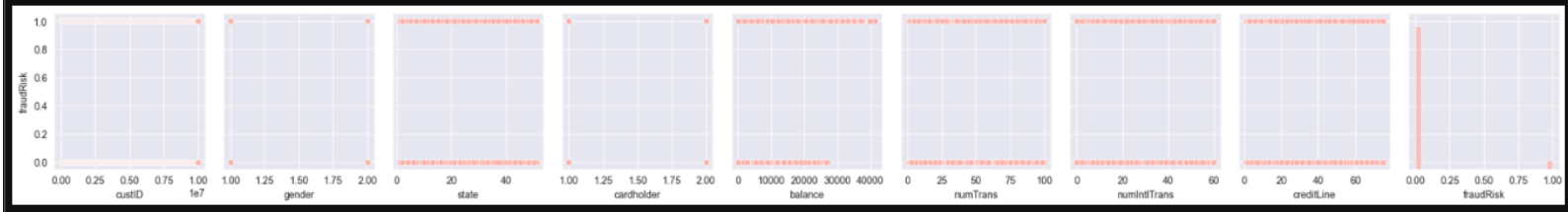


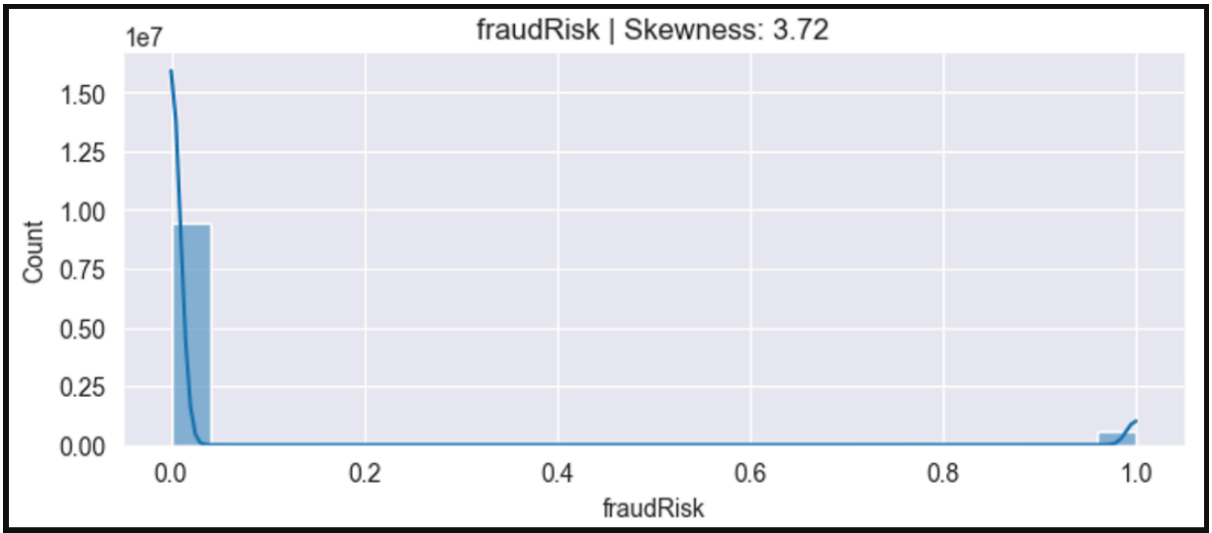
Esto podría contribuir a los falsos positivos y negativos porque de cierta forma dificulta más la detección de patrones. Una solución adecuada sería la ingeniería de características o el uso de PCA, pero nuevamente volvemos a la limitación de los tiempos de ejecución. El 60-70 por ciento de lo que ejecuté requirió al menos 20 minutos de espera, por lo que veo inviable buscar las librerías para aplicar PCA. Actualmente estoy entrenando de nuevo mi modelo debido a que quería hacerle un ajuste más, sin embargo son las 13:41 y estoy considerando la posibilidad de que no

termine de ejecutarse lo que se tiene que ejecutar a tiempo, por ello mostraré los resultados del último modelo que entrené, que a decir verdad fue un fail porque justo olvidé excluir la característica objetivo original, no me percaté que luego de aplicar one-hot encoding se conservó dicha variable. Por esta razón es que veremos un sobreajuste absoluto en el último modelo que entrené.

Debido al desbalance observado en los ploteos generados decidí investigar cómo tratar las clases desbalanceadas en Python dí con la documentación de sklearn para el paquete que trabaja con SVM [1]. Ahí se habla sobre el parámetro class_weight para tratar problemas de desbalance. De igual manera encontré que el argumento stratify sirve para el desbalance también. [2]

En estas partes se hace evidente el desbalance de clases (es lógico ya que previamente leímos que la mayoría del tráfico de red es benigno y que justo las anomalías que son ataques son difíciles de tratar debido a que son pocas observaciones a diferencia del tráfico benigno).





Nota: Excluí las variables one-Hot y fraudRisk de "X" porque me generaron sobreajuste del 100% en un modelo donde olvidé quitar fraudRisk.

Referencias:

[1] https://scikit-learn.org/stable/modules/svm.html

[2] https://hatchjs.com/python-sklearn-train_test_split-stratified/

```
[51]: print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
      print("\n---------------------------------------------- \n")
      print("---------------------------------------------- \n")
      print('Training accuracy: ', svm_model.score(X_train, y_train))
      print('Testing accuracy: ', svm_model.score(X_test, y_test))
      print('SVM Model accuracy: ', np.round(accuracy_score(y_test,y_pred),6))
      print("\n ---------------------------------------------- \n")
      print("---------------------------------------------- \n")
      print('Classification Report:\n', classification_report(y_test,y_pred,target_names=['Normal','Anormal']))
```

```
Confusion Matrix:
 [[2821196       0]
 [      0  178804]]

----------------------------------------------

----------------------------------------------

Training accuracy:  1.0
Testing accuracy:  1.0
SVM Model accuracy:  1.0

 ----------------------------------------------

----------------------------------------------

Classification Report:
               precision    recall  f1-score   support

      Normal       1.00      1.00      1.00   2821196
     Anormal       1.00      1.00      1.00    178804

    accuracy                           1.00   3000000
   macro avg       1.00      1.00      1.00   3000000
weighted avg       1.00      1.00      1.00   3000000
```

Es una pena que no pueda incluir en el reporte la versión final del modelo debido a que se me acaba el tiempo para entregar la práctica, pero espero que con este reporte y mis notas del notebook se pueda ver qué dirección llevaba mi trabajo. Cuando termine

de ejecutarse el modelo final pondré en los comentarios de clase cómo fue el rendimiento del nuevo modelo.