UNIVERSIDAD MAYOR DE SAN SIMÓN
FACULTAD DE CIENCIA Y TECNOLOGÍA
CIENCIA DE DATOS y MACHINE LEARNING

PRÁCTICA DEL PRIMER PARCIAL

CIENCIA DE DATOS y MACHINE LEARNING

Integrantes:

- Canedo Muñoz Sebastián Oscar
- Fernandez VallejosJose Franklin
- Calli Rodriguez Boris Victor
- Teran Paco Ronald

Docente: Lic. Erika Patricia Rodríguez Bilbao

Cochabamba - Bolivia 2022

I. Introducción

En los inicios de la AI, se pensaba que era suficiente que se tenga un ejército de programadores que listen una conjunto gigante de reglas que sean aplicadas a datos y así obtener respuestas, este enfoque se conoce como AI simbólica. Hoy se lo considera como programación clásica donde se tiene un conjunto de reglas y datos que dan como resultado respuestas. Sin embargo, al enfrentarse a problemas más complejos como clasificación de imágenes, reconocimiento de voz y traducción de idiomas este paradigma se queda corto.

La nueva propuesta que trae el ML es una donde se tiene datos y respuestas para obtener reglas. Estas reglas pueden ser usadas en nuevos conjuntos de datos para generar respuestas.

II. Objetivos

El principal objetivo de la práctica es poder aplicar los conocimientos adquiridos sobre Preprocesamiento de datos, Algoritmos de aprendizaje y Validación de modelos a un ejemplo en concreto. En este caso las actividades anteriormente mencionadas se aplicarán a un dataset que contiene información sobre transacciones bancarias realizadas a través de internet, esta información contiene los parámetros que permiten determinar si una transacción bancaria fue fraudulenta o legítima. El objetivo secundario se podría decir que es generar un modelo de aprendizaje automático para predecir la legitimidad o fraudulencia de transacciones bancarias realizadas por internet.

III. Marco Teórico

El primer modelo que se usó es el de regresión logística dado que es una regresión de clasificación binaria,se escogió este modelo debido al target del dataset que es de legítimo y fraude es como decir 0 y 1.

El segundo modelo que se usó es del árbol de decisión para este modelo se necesita dividir el dataset en clases ya que este modelo se necesita para comparar y así poder llegar a una clasificación.

IV. Ingeniería

Descripción

Se tiene una serie de datos lo cuales se deben usar para recaudar la información necesaria para detectar si hubo fraude o no, dato los datos que se tienen se tendrá que realizar una serie de procesamientos de datos y de los cuales aquellos que sean inconsistentes serán los datos de alarma, es decir que serán quienes son los sospechosos.

Abstracción del problema

Una posible solución puede ser un pre procesamientos de los datos para poder descartar datos basura y así poder tener datos más puros y que ayuden de mejor manera al trabajo de los datos, cumpliendo con el procesamiento se buscaría las mejores variables que estén asociadas para poder trabajarlas de maneras conjunta, teniendo estos datos se podrá trabajar de manera más eficiente y eficaz sin dar cabida a datos que sean inutilizables.

Referencia de las tecnologías a utilizar

Lenguaje

Python es un lenguaje de alto nivel, interpretado, multiplataforma y de tipado dinámico. Es uno de los lenguajes más fáciles de aprender debido a su sintaxis sencilla y su filosofía de programar de forma flexible, siendo un opuesto a lenguajes como C o C++ (que ofrecen mayor rendimiento a costa de legibilidad del código). Python es utilizado en ciberseguridad, desarrollo web y sistemas embebidos, pero recientemente el interés de la comunidad en este lenguaje ha estallado por su uso en las áreas de ML. Según los resultados del Developer Survey 2019 de Stack Overflow, Python es el lenguaje que

más personas quieren aprender. Y no es sólo casualidad, porque Python tiene el ecosistema más grande de herramientas de desarrollo de IA: Numpy, Pandas, Matplotlib e incluso Tensorflow, la librería dorada de Google para hacer Deep Learning están diseñados para su uso con Python.

Herramientas

¿Qué es Collaboratory?

Colaboratorio "Colab" para abreviar, es un producto de Google Research. Permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador. Es especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación. Desde un punto de vista más técnico, Colab es un servicio de cuaderno alojado de Jupyter que no requiere configuración y que ofrece acceso sin coste adicional a recursos informáticos, como GPUs.

¿De verdad se puede usar sin coste adicional?

Sí. Puedes usar Colab sin coste adicional.

Suena demasiado bien para ser verdad. ¿Qué limitaciones tiene?link

Los recursos de Colab no están garantizados ni son ilimitados, y los límites de uso a veces varían. Estas restricciones son necesarias para que Colab pueda ofrecer recursos sin coste adicional. Para obtener más información, consulta el apartado sobre límites de recursos.

A los usuarios que quieran disfrutar de un acceso más fiable a mejores recursos puede que les interese suscribirte a Colab Pro, pero este servicio tiene un coste monetario,etc.

En Colab, los recursos se asignan dando prioridad a los casos prácticos interactivos. Están prohibidas las acciones asociadas a operaciones informáticas en bloque, las acciones que afecten negativamente a otras y las acciones orientadas a eludir nuestras políticas. No se permite hacer lo siguiente en entornos de ejecución de Colab:

- Alojar archivos, servir contenido multimedia u ofrecer otros servicios web que no estén relacionados con la computación interactiva de Colab

- Descargar torrents o compartir archivos de punto a punto
- Usar un escritorio remoto o SSH
- Conectarse a proxies remotos
- Minar criptomonedas
- Ejecutar ataques de denegación de servicio
- Craquear contraseñas
- Usar varias cuentas para eludir las restricciones de acceso o de uso de recursos
- Crear ultra falsos (deepfakes)

Librerías

Numpy

Por dentro, los algoritmos y modelos de ML hacen operaciones con los componentes básicos del álgebra lineal: matrices y vectores. Debido a que no nos interesa enfocarnos de lleno a programar estas operaciones manualmente, utilizamos Numpy, una librería de Python que nos sirve para hacer cualquier cosa matemática en el lenguaje. Además su filosofía de programación con matrices se comparte en muchas librerías, por lo que es muy recomendable darle un vistazo.

Pandas

Previo a la etapa de ML, necesitamos procesar los datos con los que entrenamos nuestro modelo. Pandas es una librería de Python con un repertorio inmenso de operaciones para manejar datos: cargado y guardado de archivos, limpieza, agregaciones, etcétera. Esta librería es un estándar en proyectos de Data Science, y no podía faltar en esta lista.

Scikit Learn

Antes de la subida de popularidad de ML, Python ya contaba con una librería en esta área. Scikit Learn utiliza Numpy para realizar proyectos de ML y, aunque no esté

optimizada para diseñar redes neuronales y otros modelos de Deep Learning (DL) según el estado del arte, sigue siendo más que excelente para entrenar modelos como Regresiones, SVMs, etcétera.

Seaborn

Seaborn es una librería de visualización de datos para Python desarrollada sobre matplotlib . Ofrece una interfaz de alto nivel para la creación de atractivas gráficas.

Actividades y Resultados

Pre-procesamiento de datos

Regresión Logística

(explicar qué técnicas utilizas, y cual la técnica de imputación de datos faltantes que se aplicó)

Se eliminaron algunas columnas que se consideraron irrelevantes dado a que son datos que no aportarían al entrenamiento.

```
df.drop(['ip_address','user_agent','email_domain','phone_number','billing_city','billing_postal',
         'billing_state','event_timestamp','applicant_name','billing_address','merchant_id','tranaction_initiate',
         'days_since_last_logon','inital_amount'],axis=1,inplace=True)
format_column_names(df)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 12 columns):
 #   Column              Non-Null Count   Dtype
```

Se escalaron con lo que es la técnica de Min-MaxScaler para un mejor entrenamiento.

```
#escalar datos
nombres_colms = dfescalar.columns
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import MinMaxScaler

dfEscalado = ColumnTransformer([('escalar',MinMaxScaler(feature_range=(0,1)),nombres_colms)])
dfEscalado
```

```
ColumnTransformer(transformers=[('escalar', MinMaxScaler(),
                                 Index(['account_age_days', 'transaction_amt', 'transaction_adj_amt',
       'historic_velocity', 'card_bin', 'currency', 'cvv', 'signature_image',
       'transaction_type', 'transaction_env', 'locale'],
      dtype='object'))])
```

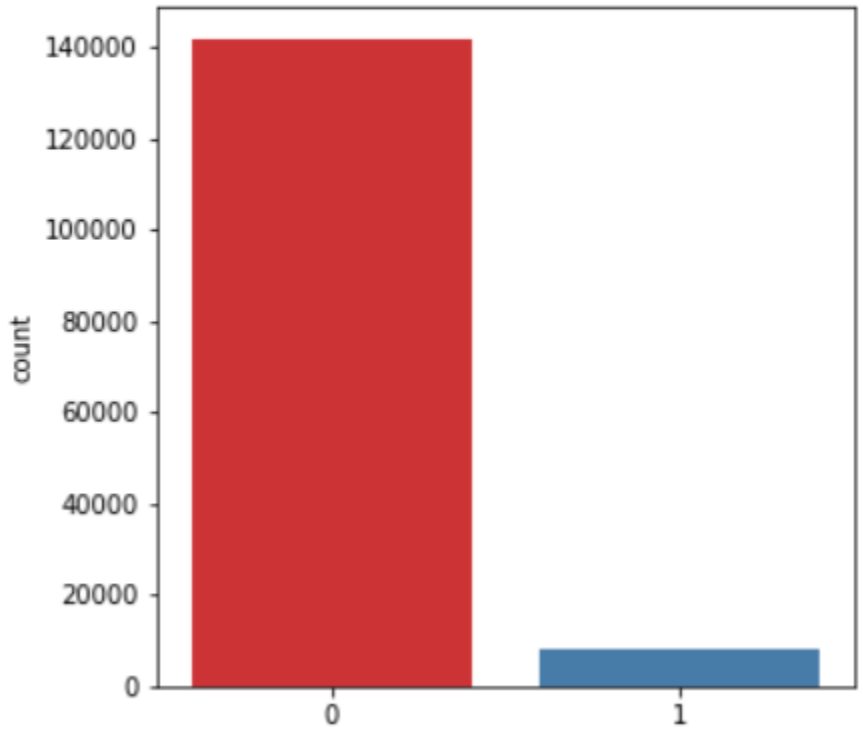
Se usó KNNImpute que usa el algoritmo de k vecinos más cercanos el cual dice que dado un k podemos clasificar un datos faltantes ya que este algoritmo busca a los vecinos que estén más cerca del dato faltante. KNNImpute funciona de la misma forma dado un k se pudo completar los datos faltantes.

```
#SE IMPUTAN LOS DATOS
imputer = KNNImputer(n_neighbors=5)
df.iloc[:,:] = imputer.fit_transform(df)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype
```

Dado al desbalance de los targets que se muestra en la figura

```
f, ax = plt.subplots(figsize=(5, 5))
ax = sns.countplot(x="event_label", data=df, palette="Set1")
plt.show()
```



se uso la técnica de agregar pesos a los targets que se repiten menos para asi poder tener una mejor clasificación una primer modelo sin aplicar la tecnica de balanceo dieron los resultados siguientes .

```
accuracy = metrics.accuracy_score(y_test,y_pred)
print("Accuracy : %s" % "{0:.3%}".format(accuracy))
```

```
Accuracy : 96.160%
```

```
precision = metrics.precision_score(y_test, y_pred)
print("Precision : %s" % "{0:.3%}".format(precision))
```

```
Precision : 80.623%
```

despues de aplicar la tecnica del balance se redujo la accuracy y la precision.

```
modelo = LogisticRegression(class_weight='balanced',max_iter=3000).fit(x_train, y_train)
```

```
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Accuracy : %s" % "{0:.3%}".format(accuracy))
```

```
Accuracy : 85.338%
```

```
precision = metrics.precision_score(y_test, y_pred)
print("Precision : %s" % "{0:.3%}".format(precision))
```

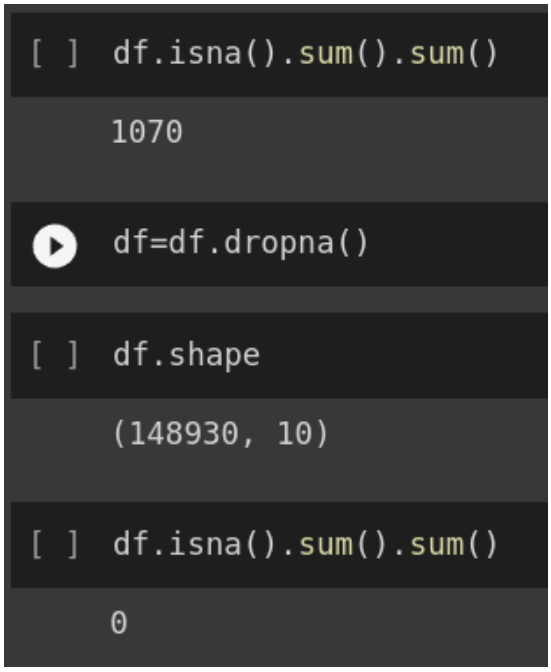
```
Precision : 25.123%
```

Árbol de clasificación

Para el preprocesamiento de los datos del árbol de clasificación se eliminaron las columnas que se sospecha que no son parte de los parámetros definitivos para la predicción o que se cree que no aportan a la predicción final. Ejemplos de estas columnas son la dirección IP o el número de teléfono o el dominio del email entre otros. En lugar de eliminar estas columnas se procedió a conservar las que sí se creía que aportan con datos importantes para la predicción, datos como el monto de transacción o la edad de la cuenta, etc en total se conservaron 10 columnas utilizando corchetes como es habitual en estructuras de tipo arreglo y asignando este mismo a una variable con el mismo nombre

```
df = df[['account_age_days', 'transaction_amt', 'transaction_adj_amt',
'historic_velocity', 'card_bin', 'currency', 'cvv', 'transaction_type',
'inital_amount', 'EVENT_LABEL']]
```

También se procedió a eliminar los datos nulos siendo estos menos del 10% del total de los datos se procedió a su eliminación utilizando la función dropna().



También se procedió a convertir las columnas que contienen tipos de datos categóricos a numéricos utilizando la fucion get_dummies() de pandas sobre las columnas que requieran de esta transformación. En este caso con las columnas que se eligieron la transformación se realizó en las columnas de 'currency', 'cvv' y 'transaction_type'. Lo que generó un nuevo dataset con 62 nuevas columnas.

```
[ ] X = pd.get_dummies(df, columns=['currency', 'cvv', 'transaction_type'])
    X.shape

    (148930, 62)
```

Determinación de datos (Features y Target)

Regresión Logística y Árbol de clasificación

Utilizando la función info() del dataset nos damos cuenta que se tienen 15000 filas o registros y 26 columnas. Inspeccionando con más detalle las columnas nos damos cuenta que existe la columna 'EVENT_LABEL' esta columna contiene dos datos categóricos "legit" y "fraud" que sería la etiqueta que nos permite diferenciar entre

transacciones legítimas y transacciones fraudulentas. Una vez que sabemos cuales son nuestras variables X independientementes (features) y nuestra variable 'y' dependiente (target o label) podemos pasar al preprocesamiento de los datos para extraer los parámetros más significativos que determinan si una transacción es o no legítima o fraudulenta.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 26 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   account_age_days      149892 non-null  float64
 1   transaction_amt       149870 non-null  float64
 2   transaction_adj_amt   149886 non-null  float64
 3   historic_velocity     149885 non-null  float64
```

Determinación de datos entrenamiento del modelo

Regresión Logística y Árbol de clasificación

Para la división de datos de entrenamiento y datos de prueba se utilizó la función train_test_split() que provee sklearn. Los datos se distribuyeron en 80% de los datos para entrenamiento y 20% para pruebas.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,)
```

Validación con al menos 5 métricas, justificar los resultados

Regresión Logística

(generar el mejor ajuste sin causar overfitting)

Exactitud de clasificación = 85.34%
Error en la clasificación =14.66%
Precisión =85.34%
Especificidad : 25.1231%

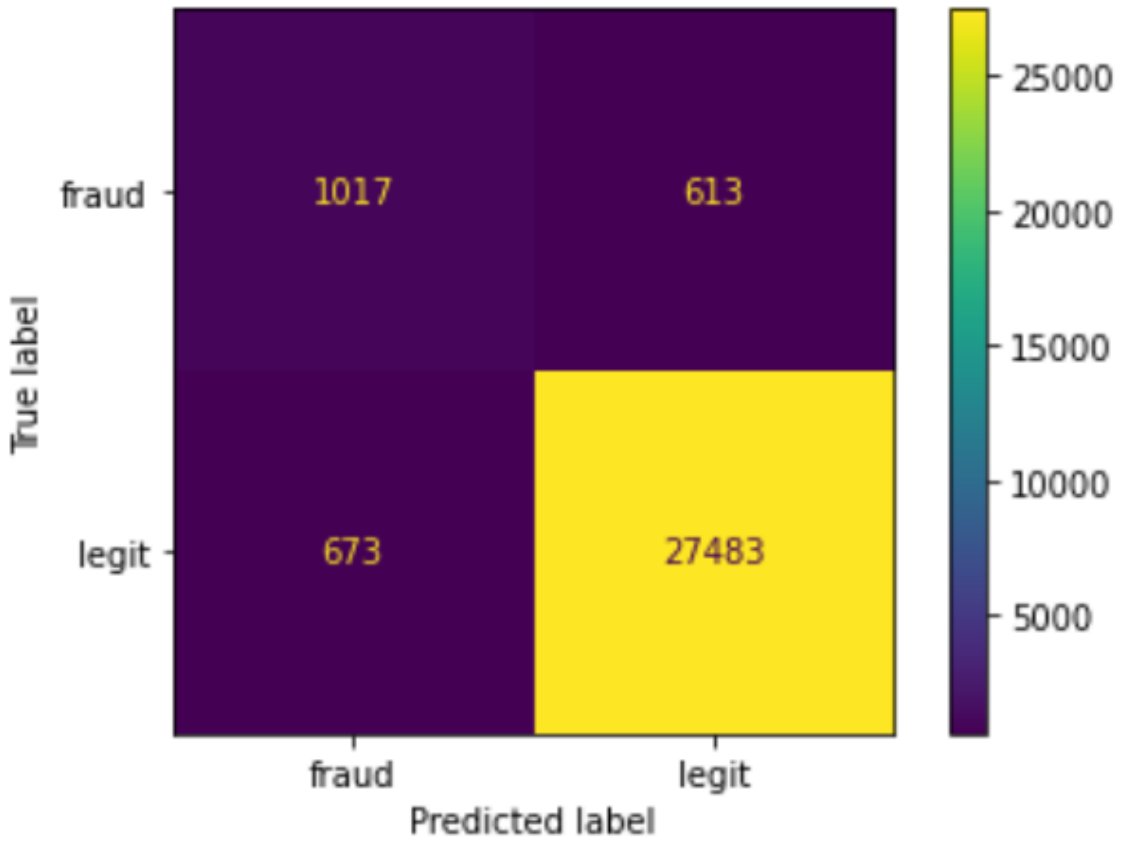
Sensibilidad (Recall): 99.0102%
Puntaje F1 : 2.0

Exactitud: El modelo puede decir clasificar con exactitud en 85.34% los fraudes
Error de clasificación: El modelo solo tiene un 14.66% de error en clasificar las transacciones que son fraudulentas y legitimas.
Precisión: El modelo tiene un porcentaje de 85.34% para identificar los fraudes.
Especificidad: El modelo tiene un porcentaje 25.1231% para clasificar correctamente los casos positivos.
Sensibilidad: La proporción de casos positivos que fueron correctamente identificadas por el algoritmo es del 99.0102%.
Puntaje F1 : 2.0 es el resumen de la precisión y la sensibilidad.

d) Realizar predicciones con datos ingresados por el usuario e interpretar los resultados(Franklin anotará en el documento)

Árbol de clasificación

Matriz de confusión



Medición	Proporción	Resultado

Exactitud y tasa de reconocimiento	$\frac{TP+TN}{P+N}$	0.9568253541932451
Tasa de error y tasa de clasificación errónea	$\frac{FP+FN}{P+N}$	0.04317464580675485
Sensibilidad, tasa positiva real, y recordatorio	$\frac{TP}{P}$	0.9781819476082004
Especificidad y tasa negativa real	$\frac{TN}{N}$	0.6239263803680981
Precisión	$\frac{TP}{TP+FP}$	0.9760974570251456

Realizar predicciones con datos ingresados por el usuario

Árbol de Clasificación

Para el siguiente ejemplo encontramos una fila del dataset que sabemos que es una transacción fraudulenta y modificamos la información para que sea similar a las transacciones fraudulentas que se encuentran en en el dataset original

10	4602.0	1587.0	32.0	5171.0	56397.0	cad	X	F	9981.0	fraud

El resultado que nos devuelve el modelo es que efectivamente es una transacción fraudulenta

V. Conclusiones

En conclusión se puede decir que para este caso en concreto de la predicción de transacciones bancarias fraudulentas o legítimas los algoritmos de Regresión Logística y Árbol de Clasificación y las técnicas de preprocesamiento utilizadas pudieron sacar resultados significativos que permiten validar el uso de estos modelos como herramienta de apoyo a la clasificación de este tipo de datos en el futuro.

Podemos decir que para el modelo de regresión logística se pudo alcanzar resultados buenos que nos facilitara la identificación de transacciones fraudulentas .Dado que el objetivo es decir si una transacción es fraudulenta o no . Se puede identificar un caso especial en el dataset que es el desbalance de objetivos(targets) esto se solventó con la técnica de aplicar peso a los datos con menor frecuencia.

Enlaces a los trabajos individuales en google colab

Arbol de decision

Regresión Logística

VI. Bibliografía

1. https://bookdown.org/victor_morales/TecnicasML/introducci%C3%B3n.html
2. https://scikit-learn.org/stable/tutorial/index.html
3. https://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/
4. https://efecode.com/preparacion-de-datos-transformacion-de-variables-categoricas-a-numericas-y-transformaciones-numericas
5. https://www.youtube.com/c/CodigoMaquina