Universidad Nacional de General Sarmiento

LABORATORIO DE CONSTRUCCIÓN DE SOFTWARE

PROYECTO PROFESIONAL 1

TRABAJO PRÁCTICO INICIAL

ENTREGA 2

INTEGRANTES

Guadalupe Nicole Arroyo

Lautaro Manuel Avalos

Federico Emanuel Farias

PROFESORES

Ing. Juan Carlos Monteros

Ing. Francisco Orozco De La Hoz

Lic. Leandro Dikenstein

INTRODUCCIÓN

En este informe daremos a conocer los pasos a seguir en la preparación de datos, las dificultades que tuvimos al momento de esta preparación y el posible entrenamiento del modelo elegido.

PREPARACIÓN DE LOS DATOS

El preprocesamiento y la limpieza de datos son tareas importantes que se deben llevar a cabo para que un conjunto de datos se pueda usar para el entrenamiento de modelos. Los datos sin procesar son a menudo ruidosos, no confiables y es posible que les falten valores, llevando al modelo a producir resultados engañosos, haciendo que todo el trabajo resulte inútil si no se detectan y corrigen antes de empezar.

Los datos de calidad son un requisito previo para los modelos predictivos de calidad. Para mejorar el rendimiento del modelo es fundamental comprobar la calidad general de los datos, de ésta manera podemos detectar problemas sobre ellos al principio, y decidir acerca de los pasos de limpieza y preprocesamiento de datos correspondientes.

Podemos comprobar la calidad general de los datos verificando el número de registros, el número de atributos (o características), los tipos de datos de atributo, el número de valores que faltan, que los datos tengan un formato correcto, y que sean coherentes.

Para preparar los datos utilizamos la librería pandas y luego, eliminamos los registros vacíos y las provincias del dataset que aparecen como otros (no nos sirve para el objetivo propuesto).

```
[ ]  # Leemos el dataset desde el repo
     url = 'https://raw.githubusercontent.com/LabSWPP12023S2G2/TPInicial/main/datasetUNC.csv'
     df = pd.read_csv(url, delimiter=';')
```

```
[ ]  # Eliminamos posibles records que esten vacíos
     df = df.dropna(axis=0)
     df.drop(df[df['PROVINCE'] == 'Otro'].index, inplace=True)
     df.drop(df[df['PROVINCE'] == 'other'].index, inplace=True)
```

Luego refinamos el dataset con las variables elegidas.

```
[ ]  # Vamos a refinar el dataset con las variables que deseamos
     variables = ['ANXIETY STATE', 'DEPRESSION', 'SUIC RISK', 'AGE', 'SUIC ATTEMPT HISTORY', 'PROVINCE']
```

```
[ ]  # Refinamos el dataset
     df_refinado = df[variables]
```

Una vez realizado el refinamiento, procedimos a dividir el dataset en un 70% para el entrenamiento del modelo y un 30% para el testeo posterior con train_test_split de sklearn.model_selection.

```
[ ]  # Dividimos el dataset, uno para entrenar y otro para testear, 70/30
     df_training, df_testing = train_test_split(df_refinado, test_size=0.3, random_state=42)

     # Guardamos los datasets!
     df_training.to_csv('training_datasetUNC.csv', index=False)
     df_testing.to_csv('testing_datasetUNC.csv', index=False)
```

Al querer comenzar el entrenamiento vimos que las variables como provincias e historial de intento de suicidio estaban en String no iban a poder formar parte del entrenamiento asi que decidimos darle un valor númerico para que puedan tener una codificación. Codificamos las provincias numéricamente, luego combinamos los datos de entrenamiento y de prueba para que así no falten datos en ninguno de los conjuntos con LabelEncoder de sklearn.preprocessing.

```
[ ]  # Codificar la columna 'PROVINCE' numéricamente
     province_label_encoder = LabelEncoder()
     all_provinces = pd.concat([df_training['PROVINCE'], df_testing['PROVINCE']])
     province_label_encoder.fit(all_provinces)
     df_training['COD_PROV'] = province_label_encoder.transform(df_training['PROVINCE'])
     df_testing['COD_PROV'] = province_label_encoder.transform(df_testing['PROVINCE'])
```

```
[ ]  # Codificar la columna 'SUIC ATTEMPT HISTORY' numéricamente
     suic_attempt_encoder = LabelEncoder()
     all_suic_attempt = pd.concat([df_training['SUIC ATTEMPT HISTORY'], df_testing['SUIC ATTEMPT HISTORY']])
     suic_attempt_encoder.fit(all_suic_attempt)
     df_training['SUIC ATTEMPT_COD'] = suic_attempt_encoder.transform(df_training['SUIC ATTEMPT HISTORY'])
     df_testing['SUIC ATTEMPT_COD'] = suic_attempt_encoder.transform(df_testing['SUIC ATTEMPT HISTORY'])
```

Una vez hecho esto, empezamos con el entrenamiento del modelo

ENTRENAMIENTO DEL MODELO

Para alcanzar nuestro objetivo es importante elegir el modelo de machine learning adecuado, ya que algunos modelos permiten predecir datos y otros clasificarlos. En nuestro caso, como necesitamos clasificar las provincias el modelo que más se ajusta es el modelo de regresión logística.

Definimos qué características nos importan del dataset.

```
[ ]  # Definir características (X) y etiquetas (y) para entrenamiento y prueba
     add_variables_x = ['ANXIETY STATE', 'DEPRESSION', 'SUIC RISK', 'AGE']
     X_train = df_training.drop(['PROVINCE', 'COD_PROV', 'SUIC ATTEMPT HISTORY', 'SUIC ATTEMPT_COD'], axis=1)
     X_train = X_train[add_variables_x]
     y_train = df_training['COD_PROV']
     X_test = df_testing.drop(['PROVINCE', 'COD_PROV', 'SUIC ATTEMPT HISTORY', 'SUIC ATTEMPT_COD'], axis=1)
     X_test = X_test[add_variables_x]
     y_test = df_testing['COD_PROV']
```

Creamos el modelo

```
[ ]  # Crear y entrenar el modelo de regresión logística
     model = LogisticRegression()
     model.fit(X_train, y_train)
```

```
[ ]  # Predecir etiquetas en los datos de prueba
     y_pred = model.predict(X_test)
```

Finalmente obtenemos los resultados

```
[ ]  # Evaluar el modelo con etiquetas específicas
     print(classification_report(y_test, y_pred, labels=unique_classes, target_names=province_label_encoder.classes_, zero_division=1))
```

```
                              precision    recall  f1-score   support

          Buenos Aires provincia       0.32      0.50      0.39        74
CABA (Buenos Aires capital)       1.00      0.00      0.00         8
                    Catamarca       0.28      0.77      0.41        74
                        Chaco       1.00      0.00      0.00        31
                      Chubut        1.00      0.00      0.00         1
                   Corrientes       0.17      0.01      0.02        87
                      Córdoba       1.00      0.00      0.00         3
                   Entre Ríos       1.00      1.00      1.00         0
                      Formosa       1.00      0.00      0.00         4
                        Jujuy       1.00      0.00      0.00         6
                     La Pampa       1.00      1.00      1.00         0
                     La Rioja       1.00      1.00      1.00         0
                      Mendoza       1.00      0.00      0.00         3
                     Misiones       1.00      0.00      0.00         5
                      Neuquén       1.00      1.00      1.00         0
                    Río Negro       1.00      0.00      0.00        11
                        Salta       1.00      0.00      0.00         5
                     San Juan       1.00      0.00      0.00         2
                     San Luis       1.00      0.00      0.00         4
                   Santa Cruz       1.00      0.00      0.00         2
                     Santa Fe       1.00      1.00      1.00         0
          Santiago del Estero       0.00      0.00      0.00         1

                    micro avg       0.29      0.30      0.29       321
                    macro avg       0.85      0.29      0.26       321
                 weighted avg       0.45      0.30      0.19       321
```

Precisión, Recall y F1-score son métricas comunes utilizadas para evaluar el rendimiento de modelos de clasificación. Estas métricas brindan información sobre cómo el modelo clasifica correctamente e incorrectamente las instancias en diferentes clases.

Podemos observar que no obtuvimos los mejores resultados en el entrenamiento y es posible que tengamos que hacer cambios en el dataset o utilizar otro tipo de aprendizaje para el objetivo que nos propusimos.

CONCLUSIONES

Pudimos tomar un mejor enfoque sobre la complejidad de crear y entrenar una inteligencia artificial que funcione de manera efectiva, en las próximas entregas intentaremos encontrar el mejor modelo o mejorar el entrenamiento para nuestro caso y así lograr el objetivo que nos propusimos.

CHANGELOG

En nuestro proyecto inicialmente optamos por la plataforma de desarrollo Anaconda Cloud debido a su potencial en el ámbito de análisis de datos y aprendizaje automático. Sin embargo, nos encontramos con dificultades al integrarlo con GitHub, lo que afectó nuestra productividad y cronograma.

Dada la complejidad técnica y la necesidad de avanzar sin obstáculos, decidimos migrar a Google Colab. Esta plataforma en la nube ofrece ventajas como acceso gratuito a GPUs/TPUs, integración con Google Drive, facilidad para compartir y colaborar en tiempo real, y eliminación de configuraciones complicadas en máquinas locales.

Creemos que este cambio nos permitirá superar los desafíos técnicos, mejorar la eficiencia y avanzar con éxito en la implementación de nuestra solución de machine learning.

BIBLIOGRAFÍA

Google Colab (https://colab.research.google.com)

Pandas documentation (https://pandas.pydata.org/docs/)

Scikit-Learn (https://scikit-learn.org/stable/)

López Steinmetz, L. C. (2021, May 3). *R Code and dataset for: Levels and predictors of depression, anxiety, and suicidal risk during COVID-19 pandemic in Argentina: The impacts of quarantine extensions on mental health state* [Dataset]. https://rdu.unc.edu.ar/handle/11086/20168