



TAREA 1 - INFORME

EXPANSIÓN DEL SISTEMA DE PREDICCIÓN DE
RECLUTAMIENTO

OBJETIVO

IMPLEMENTAR UN MODELO DE PREDICCIÓN PERSONALIZADO PARA UN SECTOR MÁS GRANDE Y AMPLIAR LA FUNCIONALIDAD DE LA APLICACIÓN.

REALIZADO POR

JOSE MANUEL ESPINOZA BONE / JOSÉ
ALBERTO RIVADENEIRA ROMERO / JORGE
ISRAEL VILLACIS OLEAS

MATERIA

CICLO DE VIDA DE LA INTELIGENCIA ARTIFICIAL

AMPLIAR EL DATASET

- Simular un dataset cinco veces al proporcionado.

Existe la constante `DATA_SIZE = 5000` que permite establecer el tamaño de los datos.

- Incluir dos nuevas características relevantes ("Languages (Ingles, Frances)" y "Certifications (de 0 a 10)").

Se colocó varios languages en la constante

```
LANGUAGES = ("English", "Spanish", "French", "German", "Chinese", "Japanese")
```

Y su respectiva probabilidad al generar los datos sintéticos en la constante

```
P_LANGUAGES = (0.4, 0.3, 0.1, 0.1, 0.05, 0.05)
```

ENTRENAR EL MODELO

- Preprocesar el nuevo dataset.

Al dataset sintético al tener columnas categóricas en las cuales los datos son string, estás son convertidas a un tipo categórico, mediante el tipo de datos de pandas `pandas.CategoricalDtype` se crea una instancia de tipo de dato categórico, usando la respectiva lista categórica, por ejemplo para Gender se utiliza la lista de géneros que se encuentra en la constante `GENDERS`, quedando como resultado `pd.CategoricalDtype(categories=GENDERS, ordered=True)`.

Luego de haber creado el tipo se convierte la columna de texto al tipo categorical antes creado, luego se obtiene el valor numérico de esa categoría y finalmente se convierte a entero, logrando de esta manera el objetivo de convertir a números las categorías.

- Entrenar el modelo de clasificación con el algoritmo RandomForest.

Se creó una función que realiza las instrucciones para entrenar el modelo usando el algoritmo RandomForest y guardar en archivo el resultado del entrenamiento. La función es `FitModel()`

ACTUALIZAR LA APLICACIÓN

Se utilizó la lib PySide que es Qt for Python, esta lib permite utilizar Qt con Python y es una lib bien potente para realizar interfaces gráficas.

La App consiste en un script de Python que captura los parámetros enviados a la app(Python.exe -m App <parámetro>), se tiene varios argumentos/parámetros:

- Install, permite instalar las dependencias necesarias para ejecutar esta app.
- Info, muestra la información del ambiente virtual actual.
- Create, crea el ambiente virtual.
- Activate, imprime el commando para activar el ambiente virtual.
- Fit, permite entrenar el modelo y guardar los resultados en archivo, adicional muestra las métricas del modelo como accuracy, f1, recall.

Para que funcione bien el script de Python ya que debe existir el ambiente virtual primero, luego existir las dependencias y finalmente ejecutar la app gráfica.

Se debe ejecutar las siguientes instrucciones en orden estricto en el cmd

```
python -m App --Create
```

```
type nul > ".venv\.\nosync"
```

```
call ".venv\Scripts\activate.bat"
```

```
python -m App --Install
```

```
python -m App --Info
```

```
python -m App
```

o simplemente ejecutar en el cmd el script App.bat que tiene en sus instrucciones ya el orden estricto.

App.bat

Al ejecutar la app queda la siguiente interfaz

- Boton PREDICT para realizar la predicción en base a los datos ingresados.
- Botón FIT para entrenar el modelo y guardar en archivo.

Recruitment Model App - Test

Age (18-50):

18 years

Experience (0-30):

0 years

Gender:

Male

Education Level:

High School

Language:

English

Certifications (0-10):

0 certifications

Click the button to predict hiring status!!!


FIT MODEL

PREDICT

- Añadir una funcionalidad para mostrar al usuario la probabilidad de la predicción (en lugar de solo mostrar "Hired" o "Not Hired").

En la app muestra el resultado de la predicción para los datos ingresados, la probabilidad de esa predicción y los datos utilizados

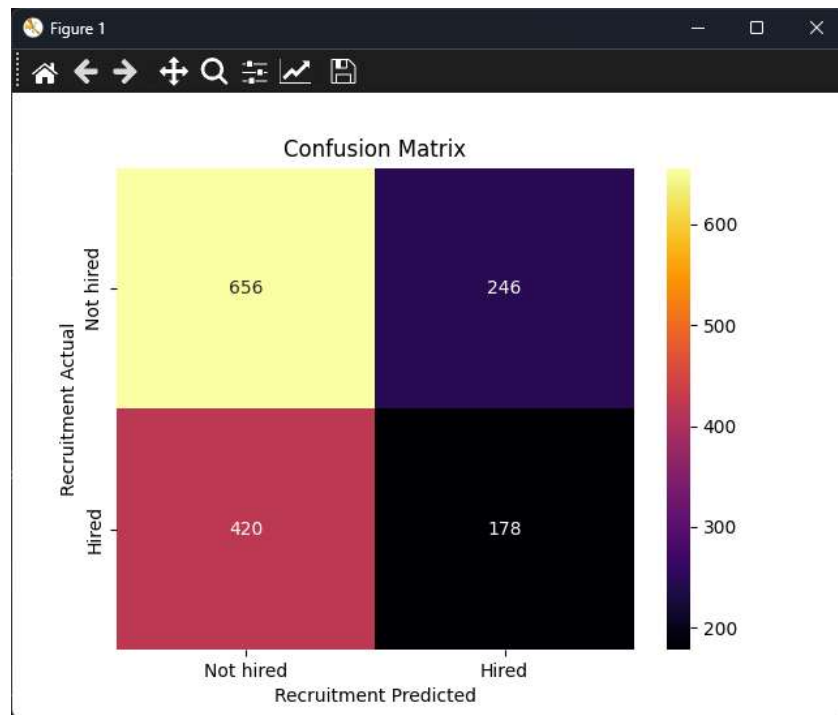
Prediction Result

 Prediction: Hired | Probability(Prediction): 0.6317

Age: 18, Experience: 0, Gender: Male, Education Level: High School, Language: English, Certifications: 0

OK

- Incluir gráficos que muestren las métricas de rendimiento de los modelos (precisión, recall, F1-score).



```


i Evaluating the model.
■ Confusion Matrix
          Predicted Not hired Predicted Hired
Actual Not hired          656          246
Actual Hired             420          178
■ Accuracy score - Report
0.556
■ Classification Report
          precision    recall  f1-score   support

     0       0.61      0.73      0.66       902
     1       0.42      0.30      0.35       598

 accuracy          0.56      1500
 macro avg         0.51      0.51      0.51      1500
 weighted avg      0.53      0.56      0.54      1500

```


CAPTURAS PREDICCIONES

 Prediction Result

Prediction: Not Hired | Probability(Prediction): 0.6600

Age: 18, Experience: 0, Gender: Male, Education Level: High School, Language: English, Certifications: 0


OK

 Prediction Result

Prediction: Not Hired | Probability(Prediction): 0.6100

Age: 30, Experience: 7, Gender: Female, Education Level: PhD, Language: Spanish, Certifications: 0


OK

 Prediction Result

Prediction: Hired | Probability(Prediction): 0.6300

Age: 35, Experience: 7, Gender: Male, Education Level: PhD, Language: German, Certifications: 6

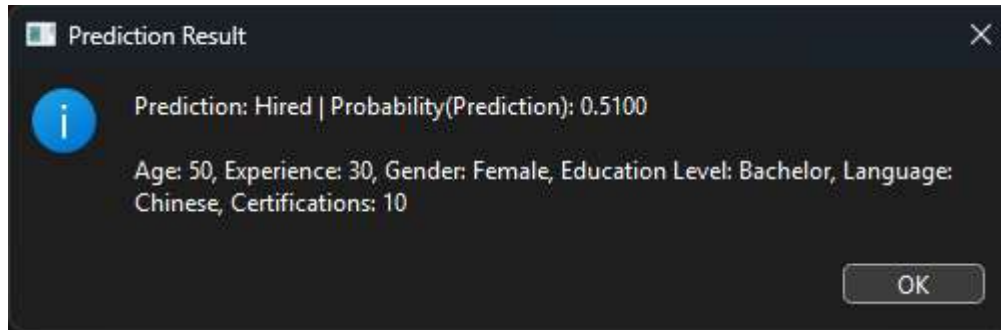
OK

 Prediction Result

Prediction: Not Hired | Probability(Prediction): 0.6500

Age: 35, Experience: 7, Gender: Female, Education Level: PhD, Language: Spanish, Certifications: 10

OK



APORTES PROPIOS

- Función para ejecutar comandos

```
def RunCommand(commandList: list[str], printCommand: bool = True, printError: bool = True) -> subprocess.CompletedProcess:
    print("⚡", " ".join(commandList))
    stdoutOutput = None if printCommand else subprocess.DEVNULL
    errorOutput = None if printError else subprocess.PIPE
    result = subprocess.run(commandList, stdout=stdoutOutput, stderr=errorOutput, text=True)
    if result.returncode != 0 and printError:
        print(result.stderr)
    return result
```

- Instalar las dependencias desde el script

```
def InstallDeps():
    print("📦 Installing deps.")
    RunCommand([sys.executable, "-m", "pip", "install", "--upgrade", "pip"], printCommand=True)
    RunCommand([sys.executable, "-m", "pip", "install", *LIBS], printCommand=True)
```

- Crear el ambiente virtual desde el script

```
def CreateVirtualEnv():
    print(f"📁 Creating Virtual Environment.")
    venvPath = Path(".venv")
    if not venvPath.exists():
        print(f"Path: {venvPath.resolve()}")
        RunCommand([sys.executable, "-m", "venv", str(venvPath)])
    else:
        print("Virtual environment already exists at:", venvPath.resolve())
```

CONCLUSIONES

- Al ser datos sintéticos es complicado tener un patrón para detectar la contratación y poder realizar las screenshots, ya que tocaba jugar a la suerte con los campos al ingresarlos y obtener las predicciones.

- Al tener datos sintéticos no se refleja un patrón de la vida real, ya que los campos del dataset son generados al azar y no concuerdan con un escenario de la vida real.
- Utilizar Qt for Python es sencillo pero requiere tiempo para especializarse en su uso, las interfaces son mucho más elegantes y elaboradas que las utilizadas usando Tkinter.

RECOMENDACIONES

- Mejorar el script de Python.
- Crear el script equivalente a App.bat pero para sistemas basados en UNIX(GNU/Linux y MacOS).

ANEXO

El código fuente se puede encontrar en el siguiente repo

[UIDE-Tareas/2-Ciclo-Vida-Inteligencia-Artificial-Tarea1](#)