Proyecto de Predicción de Depresión en Estudiantes

Introducción

Este proyecto tiene como objetivo desarrollar un modelo de inteligencia artificial capaz de predecir si un estudiante sufre de depresión a partir de diversas características. Para ello, se ha utilizado un enfoque basado en Machine Learning, aplicando técnicas de análisis exploratorio de datos (EDA), ingeniería de características, entrenamiento de modelos (Google Colab) y despliegue mediante FastAPI y PyCharm.

Caso de Uso

La salud mental de los estudiantes es un tema de gran importancia, especialmente en el contexto académico donde factores como el estrés financiero, la carga de trabajo y la falta de sueño pueden influir en el bienestar emocional. Este proyecto tiene como objetivo desarrollar un modelo de aprendizaje automático capaz de predecir si un estudiante está en riesgo de depresión basándose en diversos factores. Para ello, se ha utilizado un dataset con múltiples características relevantes, las cuales han sido analizadas y transformadas para optimizar el rendimiento del modelo.

Este sistema puede ser utilizado por instituciones educativas, psicólogos y orientadores estudiantiles para detectar tempranamente posibles casos de depresión en estudiantes. A través de un sistema automatizado basado en machine learning, se pueden identificar patrones en los datos de los estudiantes y proporcionar una evaluación de riesgo. Esto permitirá que las instituciones implementen estrategias de apoyo personalizadas, como asesoramiento psicológico o programas de bienestar, para mejorar la calidad de vida de los alumnos y reducir los impactos negativos en su rendimiento académico.

Exploración y Análisis de Datos (EDA)

Se realizó un análisis exploratorio de datos (EDA) para entender la distribución de las variables y su relación con la variable objetivo (depresión). Durante este proceso se detectaron valores faltantes, la presencia de variables categóricas y posibles correlaciones entre variables.

Carga del Archivo

- El conjunto de datos utilizado en este proyecto se encuentra en el archivo StudentDepressionData.csv, el cual contiene información sobre estudiantes y su estado de depresión.

Análisis y Visualización con AutoViz

Se ha utilizado la herramienta AutoViz para analizar la distribución de las variables, la correlación entre ellas y otros patrones relevantes, generando gráficos y visualizaciones útiles.

- Se identificó que la columna "Depression" es la variable objetivo y presenta un desbalance en las clases.
- Se analizaron distribuciones de variables clave como "Sleep Duration", "Financial Stress" y "Dietary Habits".
- Se identificaron valores faltantes en algunas columnas y se determinaron estrategias para su imputación.
- Se calcularon correlaciones entre las variables para determinar su relevancia en la predicción de la depresión..

Especificación de la Columna Objetivo

La columna Depression indica si un estudiante sufre de depresión (1) o no (0). Esta columna es la variable objetivo del modelo.

Ingeniería de Características

Antes de entrenar el modelo, fue necesario realizar varias transformaciones en los datos para mejorar su calidad y optimizar el aprendizaje automático. Algunas de las acciones realizadas incluyen:

- **Manejo de valores faltantes:** Se imputaron los valores faltantes en la columna "Financial Stress" con la media de la misma.
- **Eliminación de columnas irrelevantes:** Se descartaron variables con baja correlación con la depresión, tales como "ID", "Age", "Degree", "Profession", "Work Pressure", "Job Satisfaction", "City" y "Gender".
- **Codificación de variables categóricas:** Se transformaron variables como "Sleep Duration" y "Dietary Habits" en valores numéricos, asignando un identificador único a cada categoría.
- **Transformación de variables booleanas:** Hemos identificado dos columnas que también requieren transformación para facilitar el aprendizaje del modelo: "Have you ever had suicidal thoughts?" y "Family History of Mental Illness". Ambas columnas tienen solo dos valores posibles: "Yes" y "No", lo que las convierte en variables booleanas. Para optimizar el proceso de aprendizaje, se recomienda asignar valores numéricos a estas columnas, transformando "Yes" a 1 y "No" a 0. Esta conversión ayudará a que el modelo maneje mejor las variables y facilite su análisis en las etapas de entrenamiento y predicción.
- **Consideración del desequilibrio en datos categóricos:** Tras analizar las gráficas de distribución, se observa que los datos en las columnas "Degree", "Profession" y "City" presentan una distribución desigual. Este comportamiento puede indicar un desequilibrio en los datos recolectados, lo que podría impactar en la capacidad del modelo para generalizar correctamente. Por ejemplo, ciertas categorías como "Class 12" en "Degree" o "Student" en "Profession" tienen una representación significativamente mayor en comparación con otras categorías. Es fundamental considerar este desequilibrio al diseñar y ajustar el modelo de aprendizaje para evitar sesgos y mejorar la precisión de las predicciones. Se evaluará la posibilidad de realizar técnicas de balanceo o transformación de los datos según sea necesario.
- **Manejo de valores atípicos (Outliers):** Finalmente, en este paso se eliminaron valores atípicos en el conjunto de datos antes de proceder con el análisis o entrenamiento del modelo.
    - **Cálculo de Z-scores:** Se calculó el Z-score de cada valor en las columnas numéricas del conjunto de datos. El Z-score es una medida estadística que indica cuántas desviaciones estándar se encuentra un valor de la media de su columna. Si un valor tiene un Z-score elevado (mayor que 3 o menor que -3), se considera un valor atípico.
    - **Filtrado de valores atípicos:** Se filtraron las filas del conjunto de datos para eliminar los valores atípicos. Esto se hizo seleccionando solo aquellas filas cuyas columnas numéricas tienen Z-scores dentro de un rango de -3 a 3. Las filas con Z-scores fuera de este rango se consideran demasiado distantes de la media y se eliminaron del análisis.
    - **Preparación de características y objetivo:** Finalmente, se preparó el conjunto de datos para el entrenamiento del modelo, separando las características (las variables que el modelo utilizará para hacer predicciones) y el objetivo (la variable que queremos predecir, en este caso, Depression).
- **Creación de nuevas características:** Se generó una nueva variable llamada "Wellbeing Index", la cual combina la satisfacción con el estudio, el estrés financiero y la duración del sueño para ofrecer una medida cuantitativa del bienestar del estudiante.

Entrenamiento del Modelo

Se importan distintos modelos de clasificación (Naive Bayes, Árbol de Decisión, Random Forest, Regresión Logística, SVM y KNN) y métricas de evaluación (precisión, reporte de clasificación y matriz de confusión) para entrenar y evaluar el modelo.

Evaluación y Comparación de Modelos de Clasificación

Se entrenaron y evaluaron varios modelos de clasificación almacenados en un diccionario de modelos. Para cada modelo, se entrenó con los datos de entrenamiento (X_train, y_train) y se generaron predicciones (y_pred) sobre los datos de prueba (X_test). Luego, se imprimió un informe de clasificación (classification_report), que incluye métricas como precisión, recall y f1-score, además de la matriz de confusión (confusion_matrix).

Los posibles valores obtenidos incluyen:

- **Precisión (Precision):** Proporción de predicciones correctas entre las que el modelo clasificó como positivas.
- **Recall (Sensibilidad):** Proporción de casos positivos correctamente identificados.
- **F1-Score:** Promedio armónico entre precisión y recall.
- **Exactitud (Accuracy):** Proporción total de predicciones correctas.
- **Matriz de Confusión:** Representa los aciertos y errores en la clasificación.

Estos valores permiten comparar el rendimiento de los modelos y seleccionar el más adecuado.

Cálculo de la Curva ROC

Se calculó la curva ROC para evaluar el rendimiento de los modelos. Dado que algunos modelos pueden no contar con el atributo predict_proba, se implementó una condición dentro del bucle de evaluación para asegurar que solo los modelos que lo poseen sean evaluados correctamente. Los resultados de la curva ROC se presentan tanto en formato gráfico como en la consola, permitiendo una interpretación visual y numérica de la performance de cada modelo.

Optimización del Modelo Seleccionado

Después de evaluar los modelos, se seleccionó la **Regresión Logística** como el mejor modelo para la predicción de la depresión en estudiantes. Para mejorar su rendimiento, se intentó aumentar el AUC (Área Bajo la Curva) mediante técnicas de balanceo de datos, con el objetivo de reducir el impacto del desbalance en la variable objetivo y mejorar la capacidad del modelo para generalizar los resultados.

Guardar el modelo entrenado:

- Una vez entrenado y evaluado el modelo, se guardara este modelo utilizando una librería como joblib para que pueda ser reutilizado en el despliegue sin necesidad de volver a entrenarlo.

Despliegue del Modelo con FastAPI

Para hacer accesible el modelo, se desarrolló una API en **FastAPI** y se probó en **PyCharm** debido a las limitaciones de Google Colab para ejecutar colabcode.

Código de la API

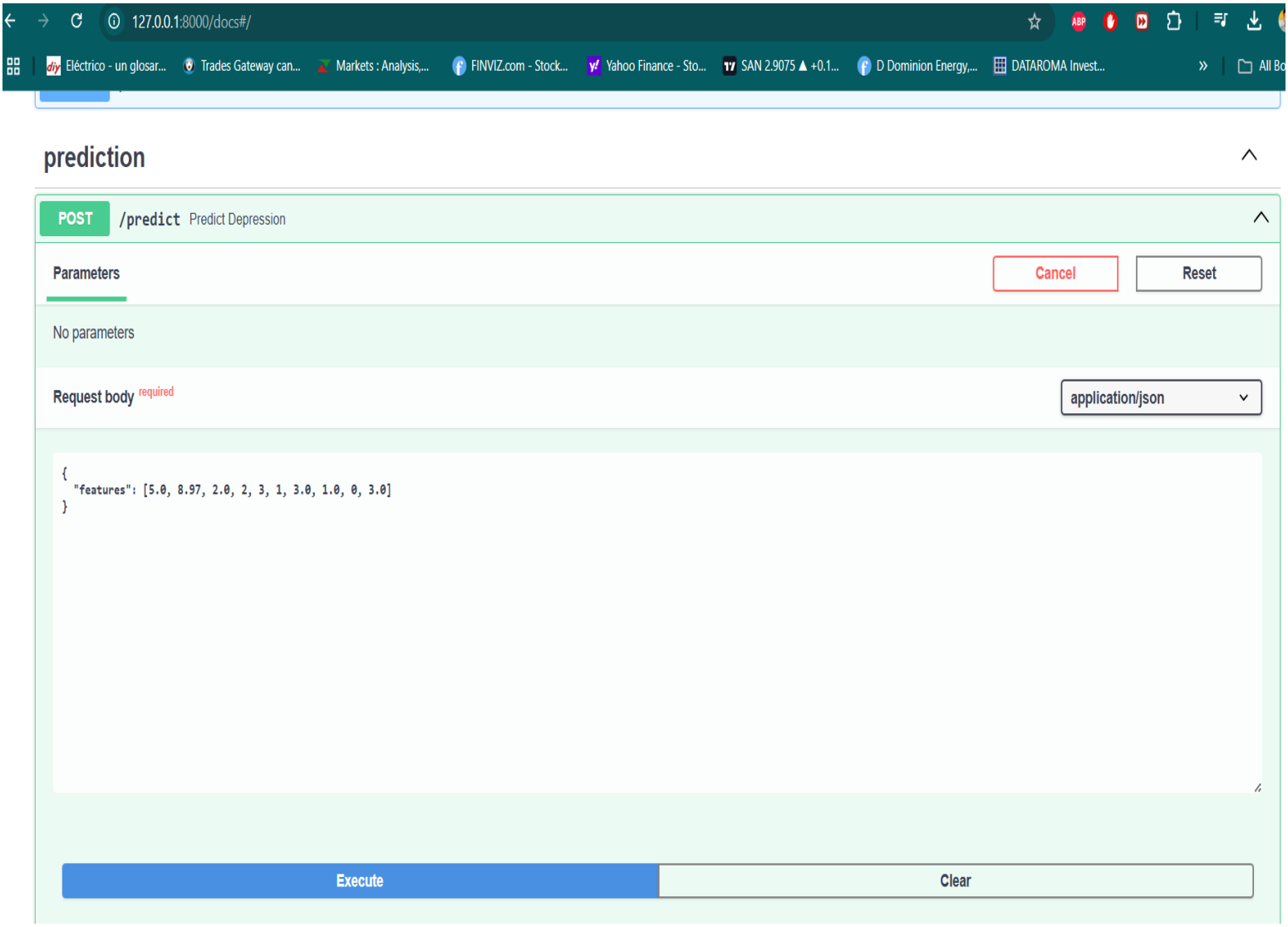
```
import joblib
from fastapi import FastAPI
from pydantic import BaseModel
import numpy as np
# Definir la estructura de los datos de entrada
class StudentData(BaseModel):
    features: list  # Lista de valores de entrada
app = FastAPI(title="Student Depression Prediction API")
```

```python
@app.get("/")
def home():
    return {"message": "¡API Student Depression Prediction está funcionando
correctamente!"}
# Cargar el modelo al iniciar el servidor
@app.on_event("startup")
def load_model():
    global model
    model = joblib.load("student_depression_model.pkl")  # Cargar el modelo guardado
@app.post("/predict", tags=["prediction"])
async def predict_depression(data: StudentData):
    try:
        # Convertir los datos en un array de numpy con el formato correcto
        features = np.array(data.features).reshape(1, -1)

        # Hacer la predicción
        prediction = model.predict(features)[0]  # Devuelve 0 o 1
        probability = model.predict_proba(features).tolist()  # Probabilidades
        # Mapear el resultado a una descripción más clara
        result_text = "En depresión" if prediction == 1 else "Sin depresión"
        return {
            "prediction": int(prediction),
            "probabilities": probability,
            "result_text": result_text,
        }
    except Exception as e:
        return {"error": str(e)}
```

Prueba del Despliegue con Postman

Para probar la API, se realizó una solicitud POST a http://127.0.0.1:8000/predict con el siguiente JSON:



Respuesta esperada:

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "features": [5.0, 8.97, 2.0, 2, 3, 1, 3.0, 1.0, 0, 3.0]
}
'
```

Request URL

```
http://127.0.0.1:8000/predict
```

Server response

Code	Details
200	Response body

```
{
  "prediction": 1,
  "probabilities": [
    [
      0.3353341198548685,
      0.6646658801451315
    ]
  ],
  "result_text": "En depresión"
}
```

Download

Response headers

```
content-length: 104
content-type: application/json
date: Thu,20 Mar 2025 20:31:14 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Conclusión

Este proyecto demuestra cómo aplicar técnicas de Machine Learning para predecir la depresión en estudiantes, utilizando análisis exploratorio de datos, ingeniería de características y entrenamiento de modelos. Finalmente, se logró desplegar exitosamente la API en un entorno local utilizando FastAPI y PyCharm, permitiendo realizar predicciones en tiempo real. Este sistema puede servir como una herramienta útil para la detección temprana de depresión en entornos educativos.