EXPRÉSATE PERÚ CON DATOS 2024

"MODELO DE CLASIFICACIÓN PARA POSTULANTES A LA UNIVERSIDAD NACIONAL DE INGENIERÍA"

PRESENTADO POR:

CARLOS ADOLFO ESPINOZA CORDOVA

ING. ESTADÍSTICA
UNIVERSIDAD NACIONAL DE INGENIERÍA

ANÁLISIS DESCRIPTIVO

- CON EL FIN DE PRESENTAR DESCRIPTIVAMENTE LOS DATOS TANTO LOS CORRESPONDIENTES A LOS POSTULANTES AL EXAMEN DE ADMISIÓN COMO A LOS POSTULANTES A TRAVÉS DEL CENTRO PRE UNIVERSITARIO DE LA UNIVERSIDAD NACIONAL DE INGENIERÍA SE EMPLEÓ EL PROGRAMA POWER BI, LO QUE NOS PERMITIÓ VISUALIZAR ADECUADAMENTE LA DISTRIBUCIÓN DE LOS DATOS.

POSTULANTES ADMISIÓN

Promedio 2021 - 2023	Promedio 2021
7.62	7.60

Promedio 2022	Promedio 2023
7.52	7.71

ESPECIALIDAD	Promedio de CALIF_FINAL
INGENIERÍA FÍSICA	9.21
MATEMÁTICA	8.76
INGENIERÍA DE TELECOMUNICACIONES	8.65
FÍSICA	8.63
INGENIERÍA MECÁNICA-ELÉCTRICA	8.57
INGENIERÍA ELECTRÓNICA	8.53
Total	**7.62**

ANIO_POSTULA

2021

2023

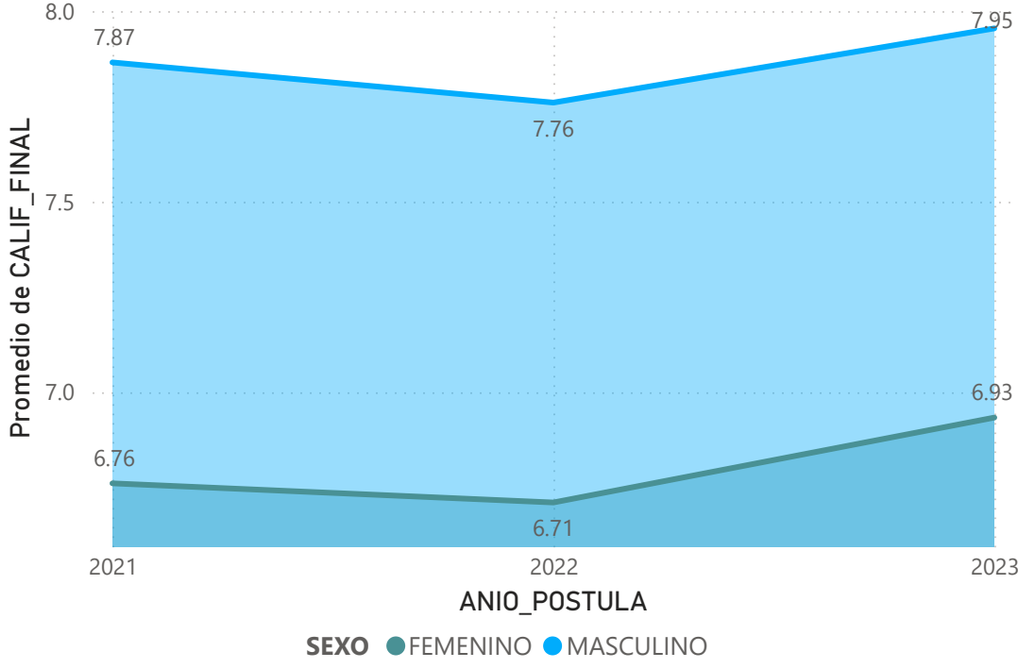
INGRESO

- [ ] NO
- [ ] SI

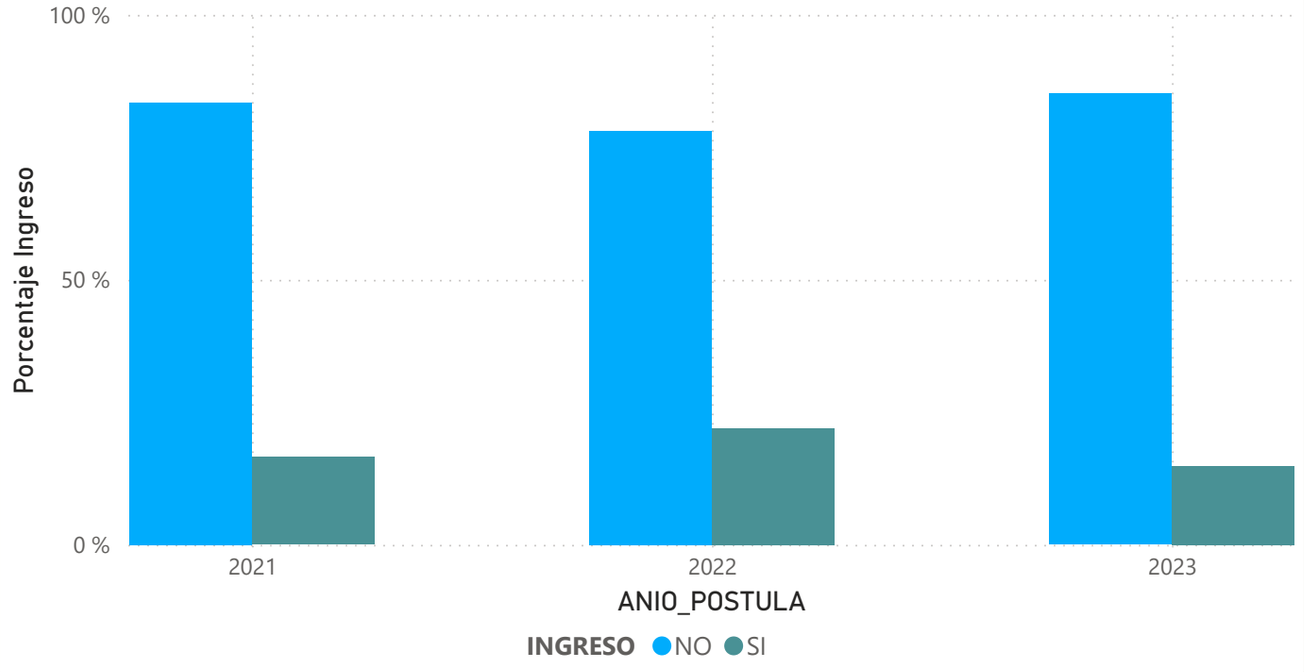
COLEGIO_ORIG...

- [ ] LIMA
- [ ] PROVINCIA

PROMEDIO DE CALIFICACIÓN



DISTRIBUCIÓN % DEL INGRESO POR AÑO DE POSTULACIÓN



POSTULANTES CEPRE UNI

Promedio 2021 - 2023	Promedio 2021
5.91	4.60

Promedio 2022	Promedio 2023
6.08	7.21

ESPECIALIDAD	Promedio de CALIF_FINAL
INGENIERÍA CIVIL	13.23
INGENIERÍA MECATRONICA	12.99
INGENIERÍA DE SISTEMAS	12.65
INGENIERÍA DE SOFTWARE	12.55
INGENIERÍA MECÁNICA	12.48
INGENIERÍA ELECTRONICA	12.26
Total	**5.91**

ANIO_POSTULA

2021

2023

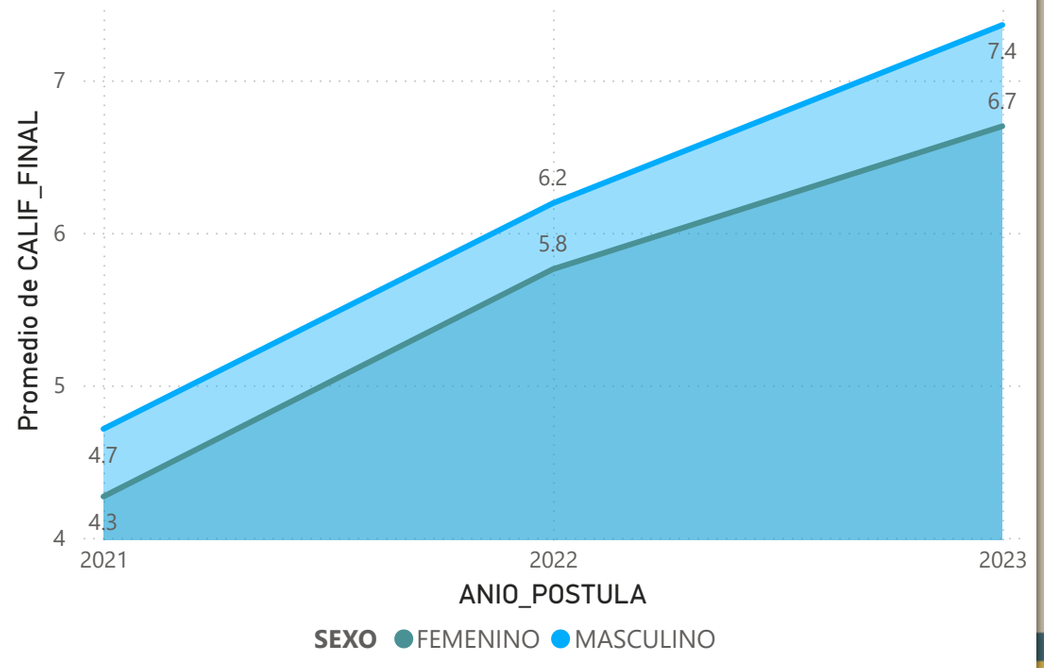
INGRESO

- [ ] NO
- [ ] SI

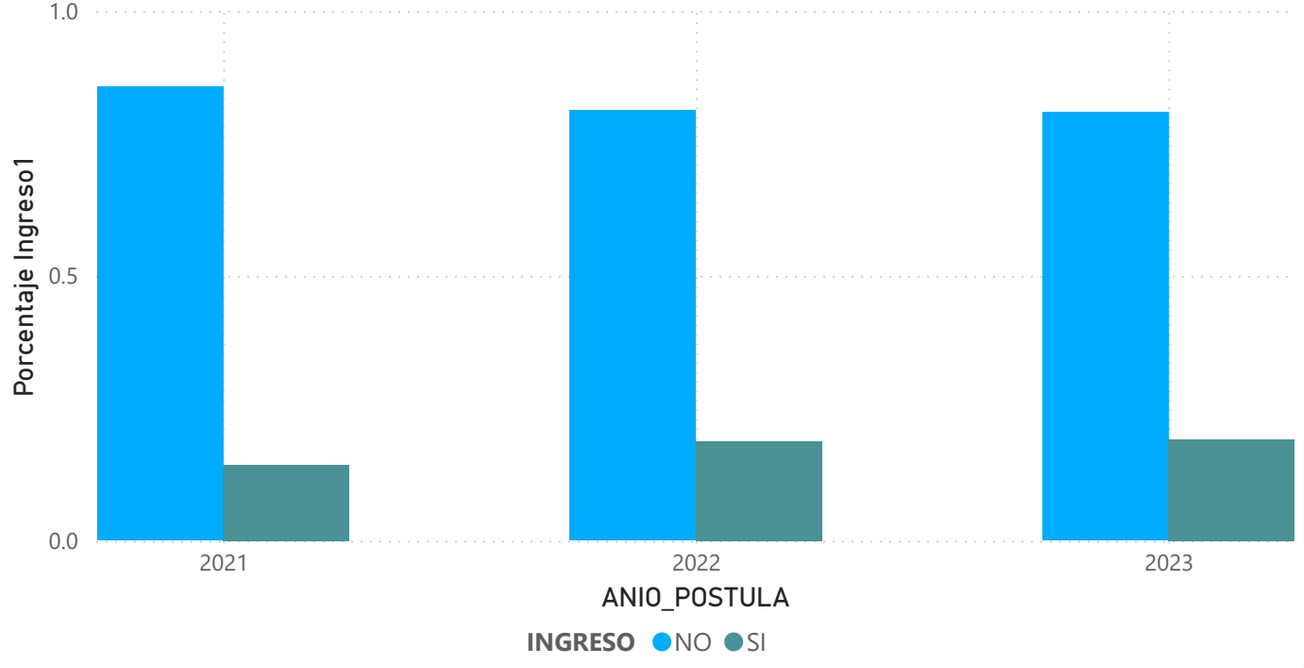
COLEGIO_ORIG...

- [ ] LIMA
- [ ] PROVINCIA

PROMEDIO DE CALIFICACIÓN SEGÚN SEXO



DISTRIBUCIÓN % DEL INGRESO POR AÑO DE POSTULACIÓN



OBJETIVO

- OBJETIVO: EL OBJETIVO DEL PRESENTE TRABAJO CONSISTIÓ EN ELABORAR UN MODELO QUE PERMITA CLASIFICAR ADECUADAMENTE A LOS POSTULANTES EN FUNCIÓN A CARACTERÍSTICAS SOCIODEMOGRÁFICAS Y ACADÉMICAS DISPONIBLES PARA PODER PREDECIR SU INGRESO A LA UNIVERSIDAD NACIONAL DE INGENIERÍA

MODELAMIENTO

- PARA SU ELABORACIÓN SE CONSIDERÓ EVALUAR DIVERSOS ALGORITMOS DE CLASIFICACIÓN ENTRE ELLOS REGRESIÓN LOGÍSTICA, ÁRBOLES DE DECISIÓN, RANDOM FOREST Y NAÏVE BAYES.

- SE TRABAJÓ CON LAS BASES DE DATOS DE LOS POSTANTES BAJO LA MODALIDAD DE EXAMEN DE ADMISIÓN Y CEPRE UNI DE LA UNIVERSIDAD NACIONAL DE INGENIERÍA.

- EL MODELO SE DESARROLLÓ EMPLEANDO EL LENGUAJE DE PROGRAMACIÓN PYTHON Y EN EL ENTORNO DE GOOGLE COLABORATORY.

CÓDIGO EMPLEADO

```
import os
import numpy as np
import pandas as pd

# Importa el archivo CSV
datos = pd.read_csv('Datos.csv', encoding='ISO-8859-1')
# Muestra los tipos de datos de cada columna
print(datos.dtypes)
```

CÓDIGO EMPLEADO

```
# Separar columnas categóricas y numéricas
categorical_cols = ['ORIGEN', 'COLEGIO_ORIGEN', 'SEXO']
numerical_cols = ['EDAD', 'ANIO_EGRESADO’]


# Importar las librerías necesarias para preprocesar
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,
OneHotEncoder
from sklearn.compose import ColumnTransformer
```

CÓDIGO EMPLEADO

```
# Crear transformador para variables categóricas y numéricas

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(), categorical_cols)
    ])

# Preprocesar los datos de entrenamiento y prueba
X_train_preprocessed = preprocessor.fit_transform(X_train)
X_test_preprocessed = preprocessor.transform(X_test)
```

CÓDIGO EMPLEADO

```
# Cargar las librerías necesarias para los modelos

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report,
accuracy_score
```

CÓDIGO EMPLEADO

```
  # Inicializar los modelos
logreg = LogisticRegression()
decision_tree = DecisionTreeClassifier()
random_forest = RandomForestClassifier()
naive_bayes = GaussianNB()

  # Creando un diccionario de modelos
models = {
    'Regresión Logística': logreg,
    'Árbol de Decisión': decision_tree,
    'Random Forest': random_forest,
    'Naive Bayes': naive_bayes
}
```

CÓDIGO EMPLEADO

```
    # Iterar sobre los modelos, entrenarlos y evaluarlos
for model_name, model in models.items():
    print(f'\n{model_name}’)

    # Entrenar el modelo

  model.fit(X_train_preprocessed, y_train)

    # Predecir con el modelo
    y_pred = model.predict(X_test_preprocessed)
```

CÓDIGO EMPLEADO

```
# Calcular la exactitud
accuracy = accuracy_score(y_test, y_pred)
print(f'Exactitud: {accuracy:.4f}')

# Reporte de clasificación
print(classification_report(y_test, y_pred))
```

CÓDIGO EMPLEADO

En base al anterior código se obtiene el siguiente resultado:

Regresión Logística
Exactitud: 0.8258

	precision	recall	f1-score	support
NO	0.83	1.00	0.90	12632
SI	0.26	0.01	0.01	2633
accuracy			0.83	15265
macro avg	0.54	0.50	0.46	15265
weighted avg	0.73	0.83	0.75	15265

CÓDIGO EMPLEADO

```
Árbol de Decisión
Exactitud: 0.8258
                precision recall f1-score support

NO                   0.83      1.00     0.90     12632
SI                   0.24      0.00     0.01      2633

accuracy                                0.83     15265
macro avg            0.53      0.50     0.46     15265
weighted avg         0.73      0.83     0.75     15265
```

CÓDIGO EMPLEADO

```
Random Forest
Exactitud: 0.8262
                  precision recall f1-score support
NO                     0.83    1.00     0.90     12632
SI                     0.35    0.01     0.02      2633

accuracy                                0.83     15265
macro avg              0.59    0.50     0.46     15265
weighted avg           0.75    0.83     0.75     15265
```

ANÁLISIS

En base a lo anterior se determina que los modelos poseen una elevada exactitud global pero el desbalance de clases que provoca que la mayoría de postulantes que no han ingresado sea clasificado adecuadamente mas no así los estudiantes que si han ingresado, lo cual se manifiesta dado que la **precisión** y el **recall** para los estudiantes que **sí** ingresan (**"SI"**) son muy bajos en todos los modelos. Los modelos tienden a predecir "NO" para la mayoría de los casos, debido a la proporción desbalanceada, lo que aumenta la exactitud global pero no es útil para predecir "SI“. Por lo cual se decide tomar en cuenta métodos de balance para la predicción.

CÓDIGO EMPLEADO

```
# Convertir variables categóricas a numéricas
from sklearn.preprocessing import LabelEncoder

# Codificar las variables categóricas
label_encoder = LabelEncoder()
X['SEXO'] = label_encoder.fit_transform(X['SEXO'])
X['COLEGIO_ORIGEN'] =
label_encoder.fit_transform(X['COLEGIO_ORIGEN'])
X['ORIGEN'] = label_encoder.fit_transform(X['ORIGEN'])
```

CÓDIGO EMPLEADO

```
# Dividir en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.3, random_state=42)
print(X_train.shape)
print(y_train.shape)
print(X_train.dtypes)
```

CÓDIGO EMPLEADO

```
from imblearn.over_sampling import SMOTE
# Aplicar SMOTE para balancear las clases en el
conjunto de entrenamiento
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote =
smote.fit_resample(X_train, y_train)
```