Índice

1 Regresión Logística

1.1 Importar las librerias

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

1.2 Importar del Dataset

```
dataset =
  ↪  pd.read_csv('https://raw.githubusercontent.com/dxnxsxn/...customers.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
dataset.head()
X[:1]
y[:1]
```

	CustomerID	Age	Gender	AnnualIncome	MaritalStatus	PurchasedInsurance
0	1	44	Male	39481	Single	0
1	2	48	Male	140583	Married	0
2	3	23	Female	70068	Married	0
3	4	33	Female	94433	Married	0
4	5	55	Male	115016	Married	1

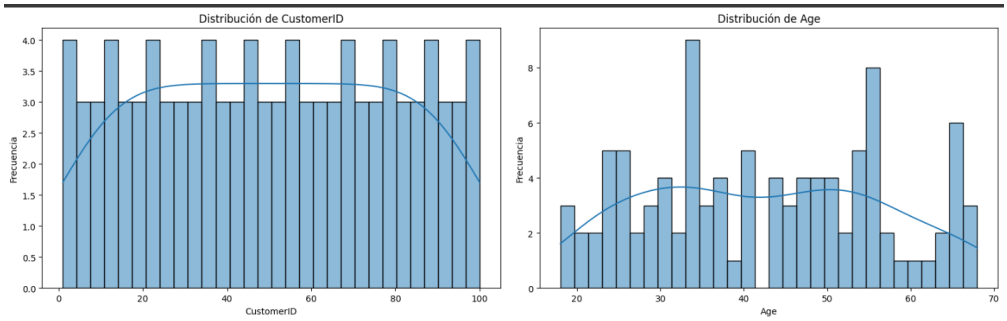
1.3 Estadísticas básicas

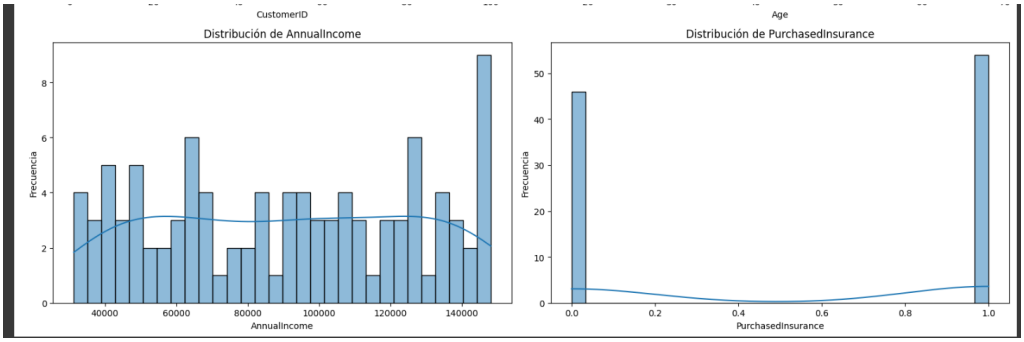
```
dataset.describe()
```

	CustomerID	Age	AnnualIncome	PurchasedInsurance
count	100.000000	100.0000	100.000000	100.000000
mean	50.500000	42.3100	90994.310000	0.540000
std	29.011492	14.1512	36169.377881	0.500908
min	1.000000	18.0000	31291.000000	0.000000
25%	25.750000	30.7500	60090.000000	0.000000
50%	50.500000	42.5000	92199.000000	1.000000
75%	75.250000	54.0000	124363.500000	1.000000
max	100.000000	68.0000	148146.000000	1.000000

1.4 Visualización de Distribuciones

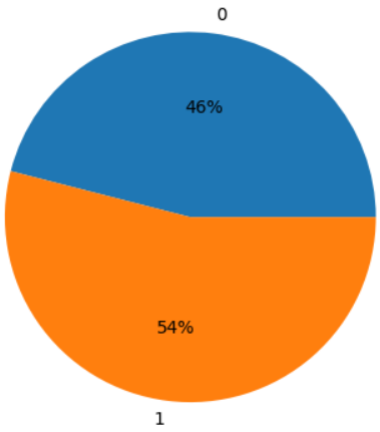
```
#Visualización de las distribuaciones
plt.figure(figsize=(16, 10))
for i, column in enumerate(['CustomerID', 'Age', 'AnnualIncome',
    'PurchasedInsurance']):
    plt.subplot(2, 2, i+1)
    sns.histplot(dataset[column], kde=True, bins=30)
    plt.title(f'Distribución de {column}')
    plt.xlabel(column)
    plt.ylabel('Frecuencia')

plt.tight_layout()
plt.show()
```





1.5 Exploración Inicial de los datos

```
# Ver que valores hay en la columna a predecir
unique, counts = np.unique(y, return_counts=True)
plt.pie(counts, labels=unique, autopct='%.0f%%')
```



1.6 Codificar las variables categoricas

```
# Para la columna Gender
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])],
    remainder='passthrough')
X = np.array(ct.fit_transform(X))

X[:1]

```

```
X=X[:, 1:] # Eliminamos la primera fila para evitar la colinealidad, se puede eliminar cualquier fila para evitarlo.
X[:1]

# Para la columna MaritalStatus
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))

X[:1]

X=X[:, 1:] # Eliminamos la primera fila para evitar la colinealidad, se puede eliminar cualquier fila para evitarlo.
X[:1]

# Para la columna MaritalStatus
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))

X[:1]

X=X[:, 1:] # Eliminamos la primera fila para evitar la colinealidad, se puede eliminar cualquier fila para evitarlo.
X[:1]
```

1.7 Dividir el dataset en conjunto de Training y conjunto de Test

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Ver que valores hay en la columna a predecir
unique, counts = np.unique(y_test, return_counts=True)
plt.pie(counts, labels=unique, autopct='%.3f%%')

X_test[:1]

y_test[:1]
```

1.8 Escalado de características

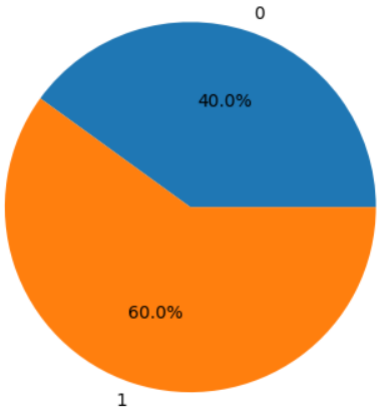
```
# Paso 1: Extraer columnas 2 y 3 y convertirlas a float
cols_to_scale = X[:, [2, 3]].astype(np.float64)  # ¡Conversión explícita necesaria!

# Paso 2: Estandarizar
scaler = StandardScaler()
scaled_cols = scaler.fit_transform(cols_to_scale)

# Paso 3: Reemplazar en el array original
X[:, [2, 3]] = scaled_cols
X[:1]
```

1.9 Entrenar el modelo de Regresión Logística con el conjunto de Training

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 23)
classifier.fit(X_train, y_train)
```

1.10 Predecir los resultados de conjunto de Test

```
y_pred = classifier.predict(X_test)
```



1.11 Predecir las probabilidades para el conjunto de Test y Training

```python
y_prob_train = classifier.predict_proba(X_train)[:,1]
y_probs = classifier.predict_proba(X_test)[:, 1]
# y_probs_test[:3]
```

1.12 Matriz de confusión y Metricas

```python
from sklearn.metrics import roc_auc_score, accuracy_score, precision_score,
↪  recall_score, f1_score, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm) # Matriz de Confución

# Calculate the AUC - ROC score
roc_auc = roc_auc_score(y_test, y_probs)

# Calculate other metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, zero_division=0)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the metrics
print(f"AUC - ROC Score: {roc_auc:.3f}")
print(f"Accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 Score: {f1:.3f}")
```