# Use Case

Alexis Michel Hernandez Robledo
Gabriela Hernandez Mandujano

# The Dataset and the Chosen Problem

Our project focuses on a personality dataset (personality_datasert.csv). This dataset contains various features that describe user behavior and interactions, such as posting frequency, attendance to social events, time online, among others.

The main problem we addressed was personality classification, specifically whether an individual tends to be Introverted or Extroverted, based on their behavioral data. In addition, we explored a regression problem, attempting to predict a continuous numerical value, such as Social_event_attendance from other characteristics.

# Modeling and Transformations Approach

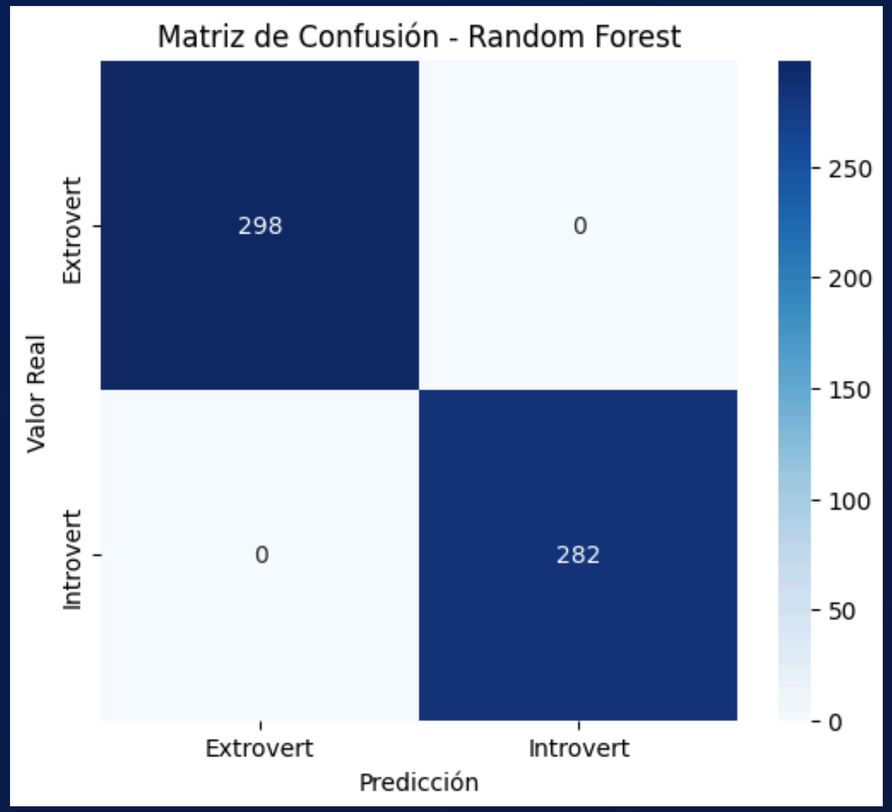
## Decision tree

An intuitive model that makes decisions by dividing data into branches, learning simple rules.

```
tree_classifier = DecisionTreeClassifier(random_state=42)
tree_classifier.fit(X_train_scaled, y_train)
y_pred_tree = tree_classifier.predict(X_test_scaled)
accuracy_tree = accuracy_score(y_test, y_pred_tree)
print(f"\nPrecisión del Árbol de Decisión: {accuracy_tree:.4f}")
print("\nReporte de Clasificación del Árbol de Decisión:")
print(classification_report(y_test, y_pred_tree, target_names=le.classes_
```

# Random Forest

A more robust method that combines multiple decision trees to obtain more accurate and reliable predictions in personality classification.




```
from sklearn.ensemble import RandomForestClassifier # Asegúrate de que esta importación e
sté al inicio del notebook

print("\n--- Modelo de Random Forest (Clasificación de Personalidad) ---")

random_forest_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest_classifier.fit(X_train_scaled, y_train)
y_pred_rf = random_forest_classifier.predict(X_test_scaled)

accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f"Precisión del Random Forest: {accuracy_rf:.4f}")

print("\nReporte de Clasificación del Random Forest:")
print(classification_report(y_test, y_pred_rf, target_names=le.classes_))

feature_importances = pd.Series(random_forest_classifier.feature_importances_, index=X.co
lumns)
print("\nImportancia de las características (Random Forest):")
print(feature_importances.sort_values(ascending=False))

plt.figure(figsize=(10, 6))
sns.barplot(x=feature_importances.sort_values(ascending=False).index, y=feature_importanc
es.sort_values(ascending=False).values, palette='viridis')
plt.title('Importancia de las Características (Random Forest)')
plt.xlabel('Características')
plt.ylabel('Importancia')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

## Linear Regression

A simple but effective model for predicting a continuous value, in our case, attendance at social events.

```
X_reg = df.drop(['Personality', 'Personality_Encoded', 'Social_event_attendance'], axis=
1)
y_reg = df['Social_event_attendance']

X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg, test_si
ze=0.2, random_state=42)

scaler_reg = StandardScaler()
X_train_reg_scaled = scaler_reg.fit_transform(X_train_reg)
X_test_reg_scaled = scaler_reg.transform(X_test_reg)

X_train_reg_scaled = pd.DataFrame(X_train_reg_scaled, columns=X_reg.columns, index=X_trai
n_reg.index)
X_test_reg_scaled = pd.DataFrame(X_test_reg_scaled, columns=X_test_reg.columns, index=X_t
est_reg.index)

linear_reg_model = LinearRegression()
linear_reg_model.fit(X_train_reg_scaled, y_train_reg)
y_pred_reg = linear_reg_model.predict(X_test_reg_scaled)

mae = mean_absolute_error(y_test_reg, y_pred_reg)
mse = mean_squared_error(y_test_reg, y_pred_reg)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_reg, y_pred_reg)

plt.figure(figsize=(10, 6))
plt.scatter(y_test_reg, y_pred_reg, color='red', alpha=0.7)
plt.plot([y_test_reg.min(), y_test_reg.max()], [y_test_reg.min(), y_test_reg.max()], colo
r='black', lw=2)
plt.xlabel("Valores Reales de Social_event_attendance")
plt.ylabel("Predicciones de Social_event_attendance")
plt.title("Valores Reales vs. Predicciones (Regresión Lineal)")
plt.grid(True)
plt.show()
```

# Comparison of Models

## For Regression (Social_event_attendance - Linear Regression):

- MAE: Tells us the mean absolute error of our predictions with respect to the actual values.
- MSE, RMSE and $R^2$ Score that tells us what proportion of the variability in social event attendance is explained by our model.

## For Classification (Personality - Decision Tree and Random Forest):

Confusion Matrix, Accuracy Precision, Recall and F1-score which give us a more detailed view of the performance by each class.

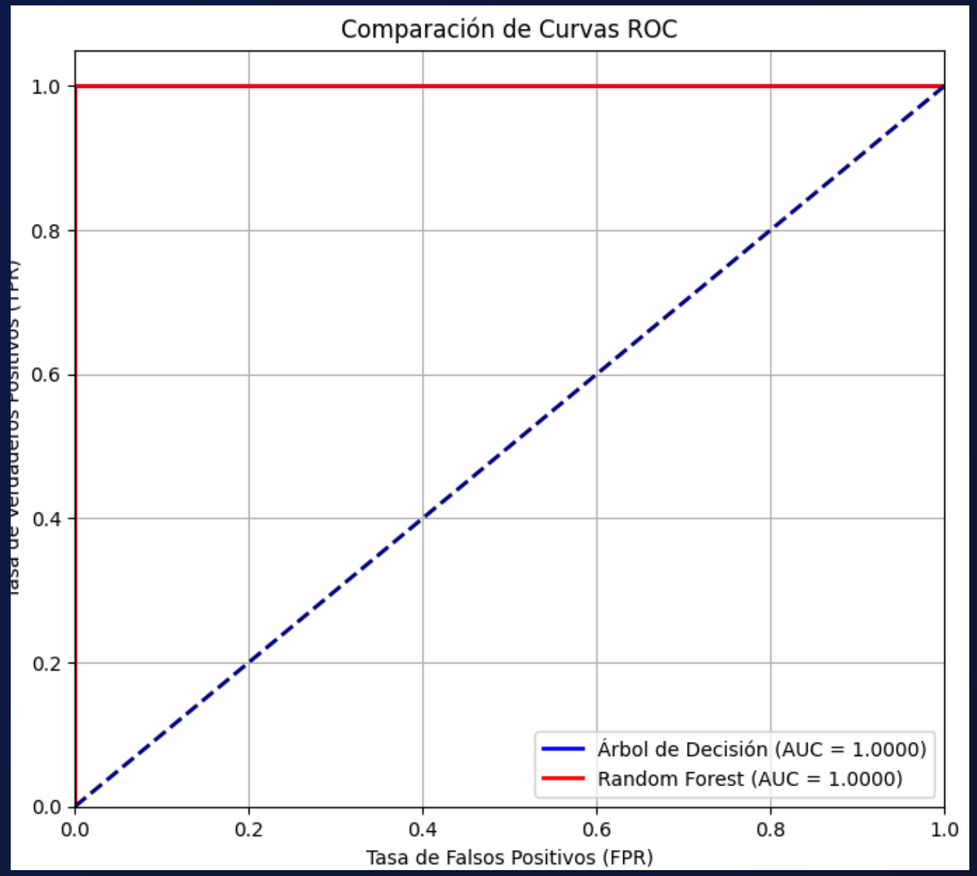
ROC AUC to assess the model's ability to distinguish between personality classes.

# Comparison of Models

## Decision Tree – Random Forest

```
plt.figure(figsize=(8, 7))
plt.plot(fpr_tree, tpr_tree, color='blue', lw=2, label=f'Árbol de Decisión (AUC = {roc_auc_tree:.4f})')
plt.plot(fpr_rf, tpr_rf, color='red', lw=2, label=f'Random Forest (AUC = {roc_auc_rf:.4f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tasa de Falsos Positivos (FPR)')
plt.ylabel('Tasa de Verdaderos Positivos (TPR)')
plt.title('Comparación de Curvas ROC')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()

print(f"Modelo              | Precisión Global | F1-Score (promedio) | ROC AUC")
print(f"--------------------|------------------|---------------------|---------")
print(f"Árbol de Decisión   | {accuracy_tree:.4f}         | (Ver Reporte)       | {roc_auc_tree:.4f}")
print(f"Random Forest       | {accuracy_rf:.4f}         | (Ver Reporte)       | {roc_auc_rf:.4f}")
```



# Thank you