DEFINICIÓN DEL MODELO PRINCIPAL

Arquitectura del Modelo

MODELO PRINCIPAL *- Random Forest Classifier*

```
model = RandomForestClassifier(
  n_estimators=200,        # 200 árboles en el ensemble
  max_depth=15,            # Profundidad máxima de 15 niveles
  min_samples_split=5,     # Mínimo 5 muestras para dividir un nodo
  min_samples_leaf=2,      # Mínimo 2 muestras en nodos hoja
  max_features='sqrt',     # Características: raíz cuadrada del total
  class_weight='balanced', # Balance automático de clases
  random_state=SEED,       # Semilla para reproducibilidad
  bootstrap=True,          # Muestreo bootstrap activado
  criterion='gini'         # Criterio de división: índice Gini
)
```

🔧 **Parámetros Optimizados**

Parámetro	Valor	Descripción
n_estimators	200	Número de árboles en el bosque
max_depth	15	Controla sobreajuste limitando profundidad
min_samples_split	5	Evita divisiones con pocas muestras
min_samples_leaf	2	Garantiza hojas con muestras suficientes
max_features	sqrt	√(n_features) para diversidad
class_weight	balanced	Compensa desbalanceo de clases

Características de Entrada

El modelo utiliza **10 características** extraídas de las imágenes:

python

```
caracteristicas_compatibles = [
'intensidad_promedio',    # Intensidad promedio de píxeles
'contraste',              # Desviación estándar de intensidades
'entropia',               # Medida de desorden/textura
'asimetria',              # Simetría de la distribución
'curtosis',               # Medida de "picudez" de la distribución
```

'densidad_bordes', *# Proporción de píxeles de borde*
'magnitud_gradiente_promedio', *# Promedio de gradientes*
'hu_momento_1', *# Momento de Hu 1 (dispersión)*
'hu_momento_2', *# Momento de Hu 2 (elongación)*
'heterogeneidad' *# Contraste/Intensidad (textura)*
]

⚙️ **Preprocesamiento de Características**

python
 # Estandarización de características
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_compatible)

 # Codificación de etiquetas
le = LabelEncoder()
y_encoded = le.fit_transform(y) *# malignant:0, benign:1, normal:2*

🎯 **Esquema de Clasificación**

python
 # CLASES DE SALIDA
CLASSES = ["malignant", "benign", "normal"]

 # ESTRUCTURA DE SALIDA
 # Probabilidades: [P(malignant), P(benign), P(normal)]
 # Diagnóstico: Clase con mayor probabilidad
 # Confianza: Máxima probabilidad obtenida

📈 **Métricas de Evaluación**

El modelo se evalúa con:

python
métricas = {
'accuracy': accuracy_score(y_test, y_pred),
'precision': precision_score(y_test, y_pred, average='weighted'),
'recall': recall_score(y_test, y_pred, average='weighted'),
'f1_score': f1_score(y_test, y_pred, average='weighted'),
'confusion_matrix': confusion_matrix(y_test, y_pred)
}

🔄 **Proceso de Entrenamiento**

python
def entrenar_modelo_completo(X, y):
"""Flujo completo de entrenamiento"""

 # 1. Preparar características

```
X_features = extraer_caracteristicas(X)

# 2. Preprocesar
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_features)

# 3. Codificar etiquetas
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# 4. Entrenar modelo
model = RandomForestClassifier(
n_estimators=200,
max_depth=15,
min_samples_split=5,
min_samples_leaf=2,
class_weight='balanced',
random_state=SEED
)
model.fit(X_scaled, y_encoded)

return model, scaler, le
```

🔍 **Características Técnicas**

- **Algoritmo**: Ensemble Learning (Random Forest)
- **Tipo**: Clasificación multiclase (3 clases)
- **Entrada**: 10 características numéricas
- **Salida**: Probabilidades para 3 clases
- **Regularización**: Parámetros de profundidad y muestras mínimas
- **Balanceo**: class_weight='balanced' para datos desbalanceados

💡 **Ventajas del Modelo Elegido**

1. **Robusto** a overfitting por parámetros conservadores
2. **Balanceado** automáticamente para datos desbalanceados
3. **Interpretable** importancia de características disponible
4. **Estable** gracias a ensemble y random state
5. **Eficiente** en entrenamiento y predicción
6.

Este modelo fue seleccionado después de optimización con RandomizedSearchCV y demostró el mejor balance entre precisión, equidad y robustez en el análisis de imágenes tiroideas.