Daniel Yaruro Contreras
Juan Toloza Parada
Oscar Ortega Lozano

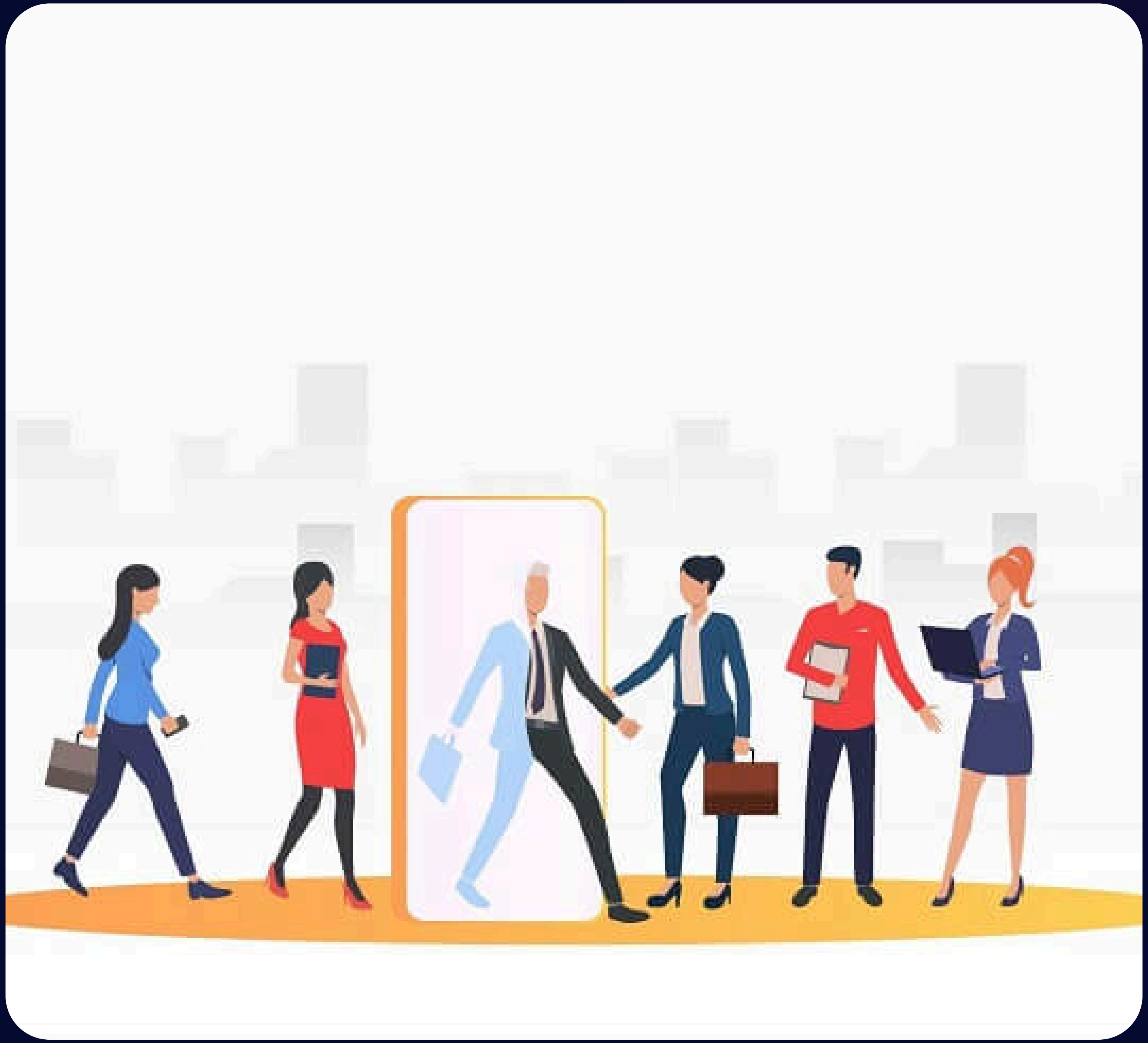
PROYECTO
MACHINE LEARNING CLASSIFICATION

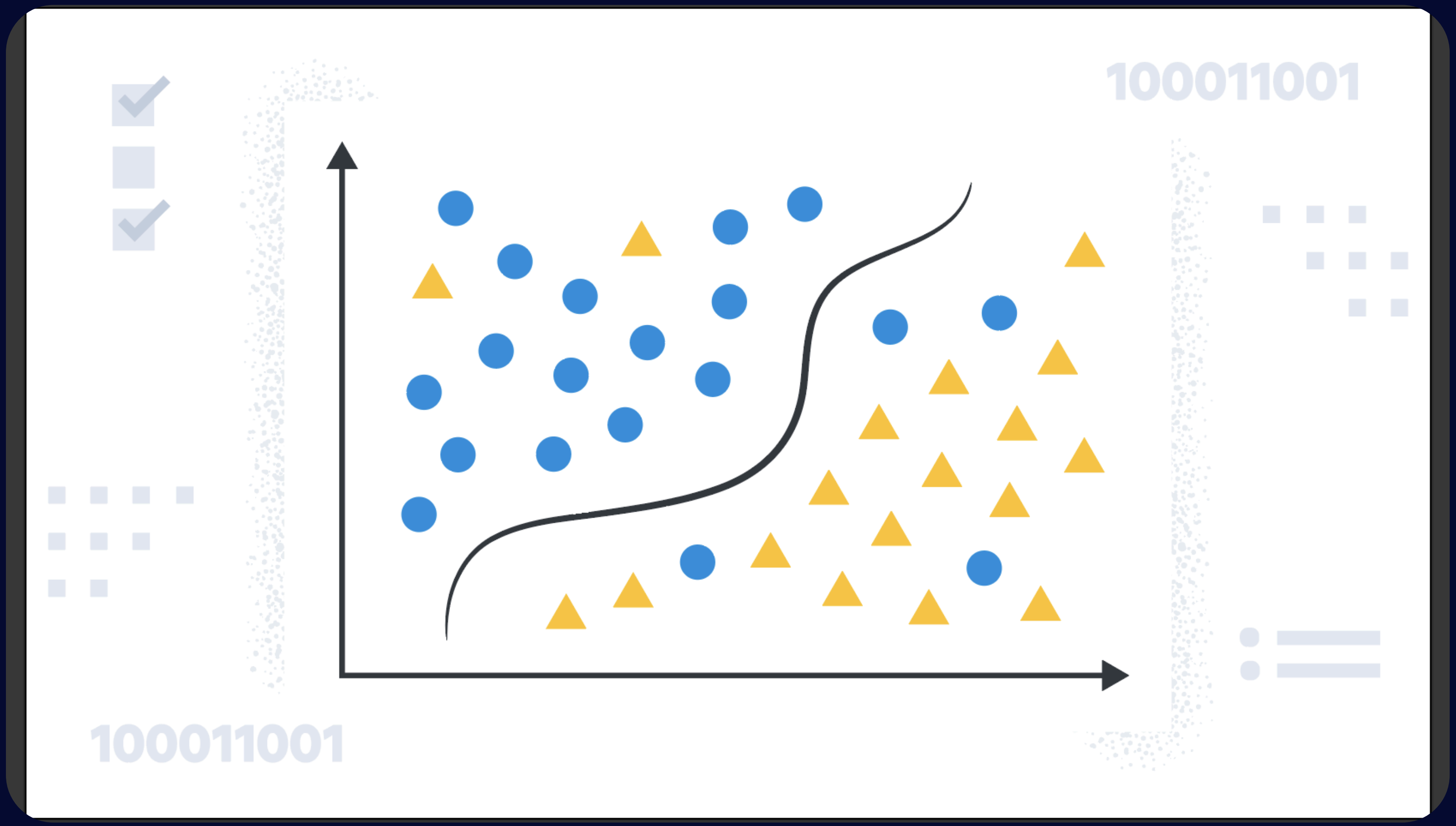
Prediction Placement Status

INTRODUCTION

PLACEMENT PREDICTION

Este dataset se centra en la predicción de la empleabilidad de estudiantes en función de su desempeño académico, experiencia práctica y habilidades blandas. Contiene información sobre calificaciones, participación en pasantías, proyectos, certificaciones y actividades extracurriculares, así como puntajes en pruebas de aptitud y entrenamientos específicos para la colocación laboral. Su objetivo es analizar qué factores influyen en la posibilidad de que un estudiante consiga un empleo tras finalizar sus estudios.





CLASSIFICATION PLACEMENT STATUS

Este campo (Placement Status) indica si un estudiante logró o no conseguir un trabajo después del proceso de reclutamiento universitario. Los valores típicos de esta columna son:

Placed

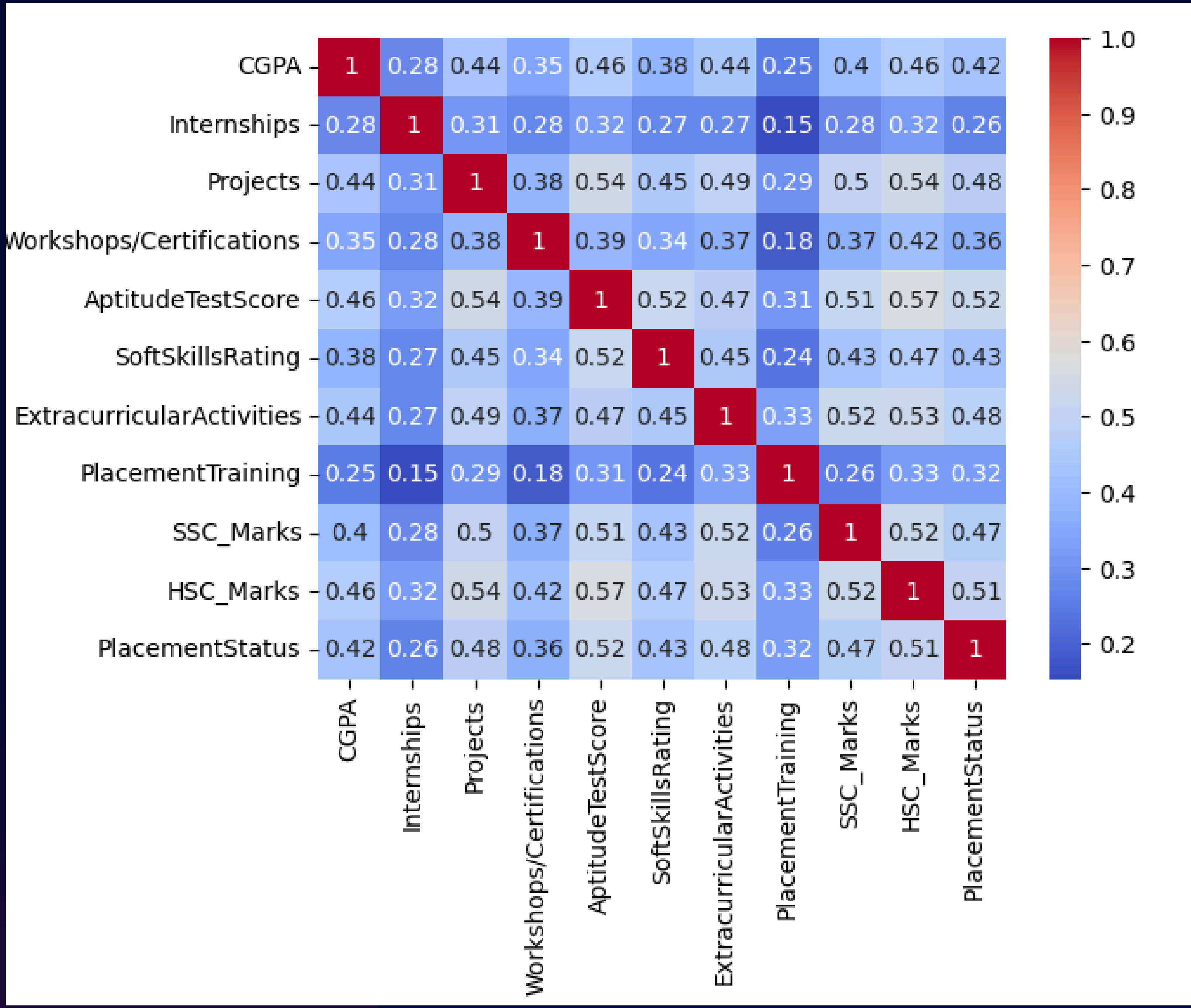
El estudiante fue contratado por una empresa.

Not Placed

El estudiante no consiguió un empleo durante el proceso de colocación.

CARACTERISTICAS RELACIONADAS

Haremos nuestro primer modelo basandonos en variables que estan mas correlacionadas con el ground truth las cuales son:



01

CGPA

Correspondientes al CGPA de cada estudiante del dataset

02

Projects

Esta columna representa la cantidad de proyectos que han hecho los estudiantes

03

Extracurricular Activities

Helps programmers keep track of code changes, collaborate on projects, and maintain consistency.

04

Aptitude Test Score

Esta columna representa la calificacion que obtuvieron los estudiantes en el test de actitud que hace la empresa o organizacion

05

SSC_Marks

Dato númerico que representa las calificaciones obtenidas por el estudiante en la educación secundaria

06

HSC_Marks

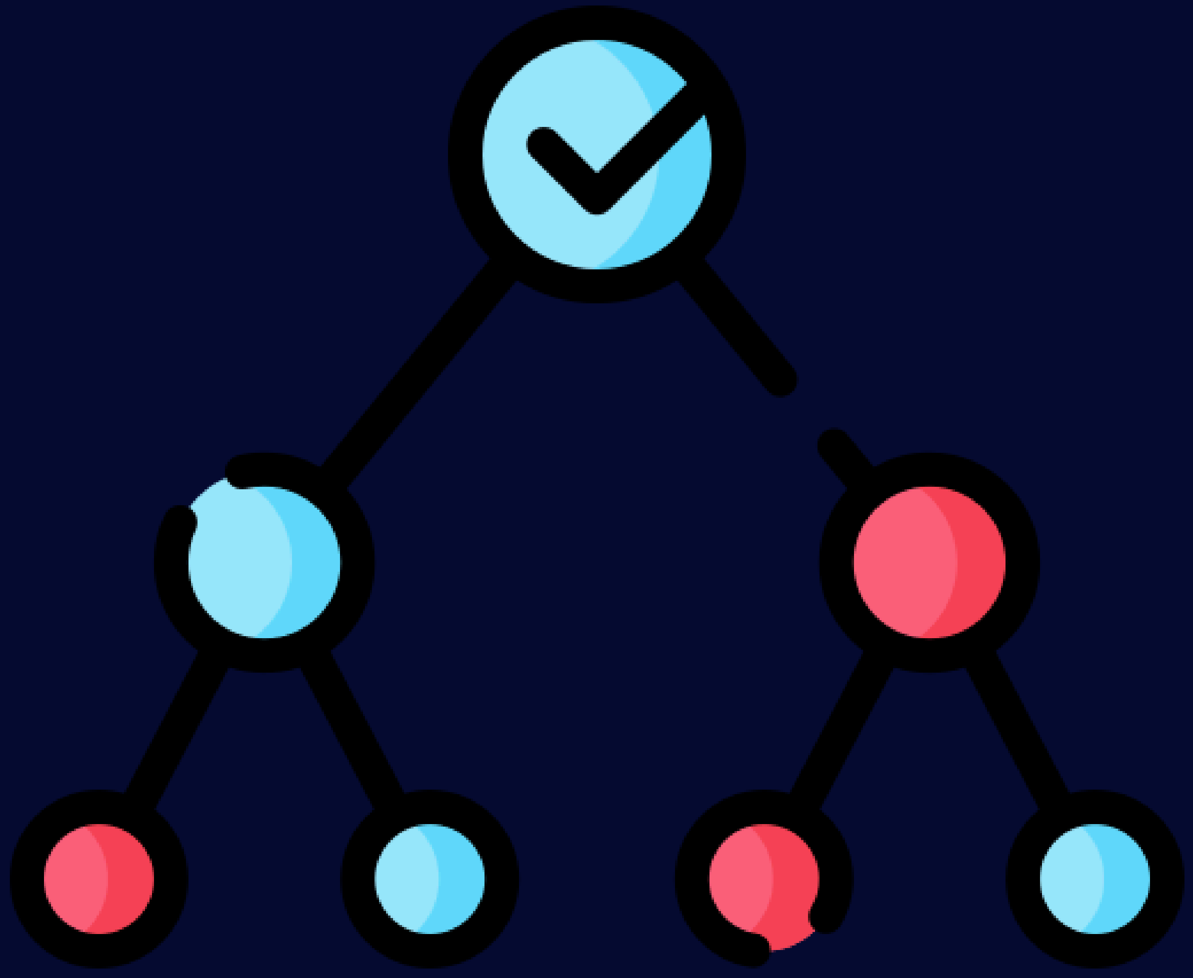
Dato númerico que epresenta las calificaciones obtenidas en la educación preuniversitaria

MACHINE LEARNING MODELS

TRAIN TEST SPLIT PROCCESS

```python
columnas_relevantes = ["CGPA",
  "Projects",
  "ExtracurricularActivities",
  "AptitudeTestScore",
  "SSC_Marks",
  "HSC_Marks"]
X = df[columnas_relevantes]
y = df["PlacementStatus"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

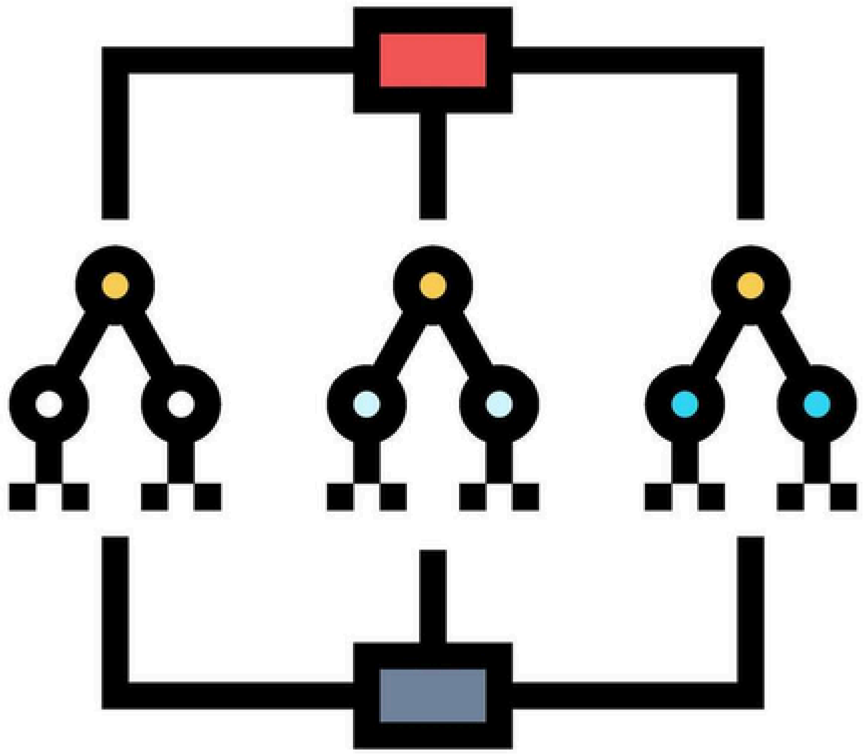
```
El tamaño del X_train es:  (8000, 6)
El tamaño del y_train es:  (8000,)
El tamaño del X_test es:  (2000, 6)
El tamaño del y_test es:  (2000,)
```



DESICION TREE

```python
#@title Desicion Tree
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

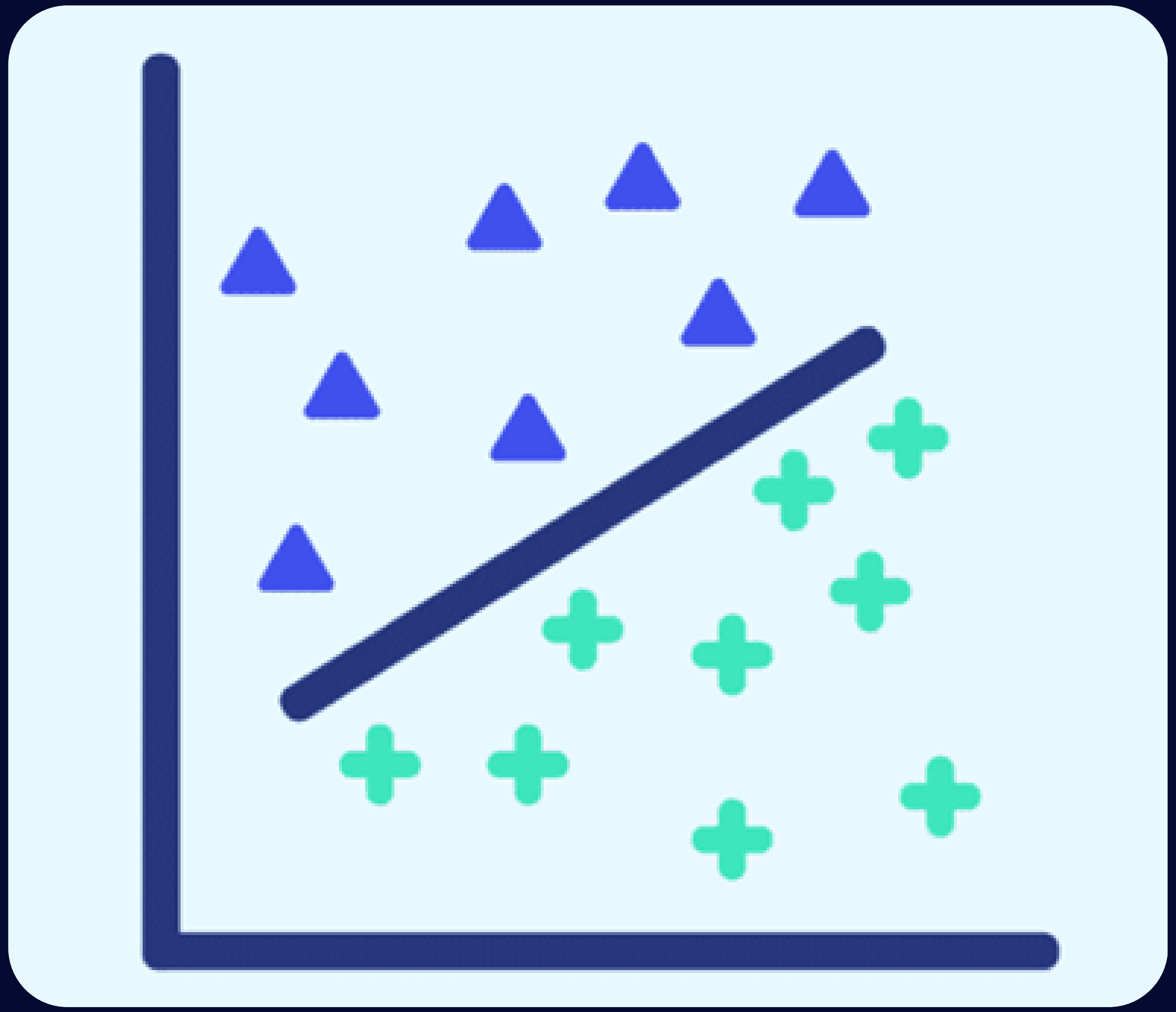
```
Accuracy: 0.7245
```



RANDOM FOREST

```python
#@title Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.772
```



SUPPORT VECTOR MACHINE

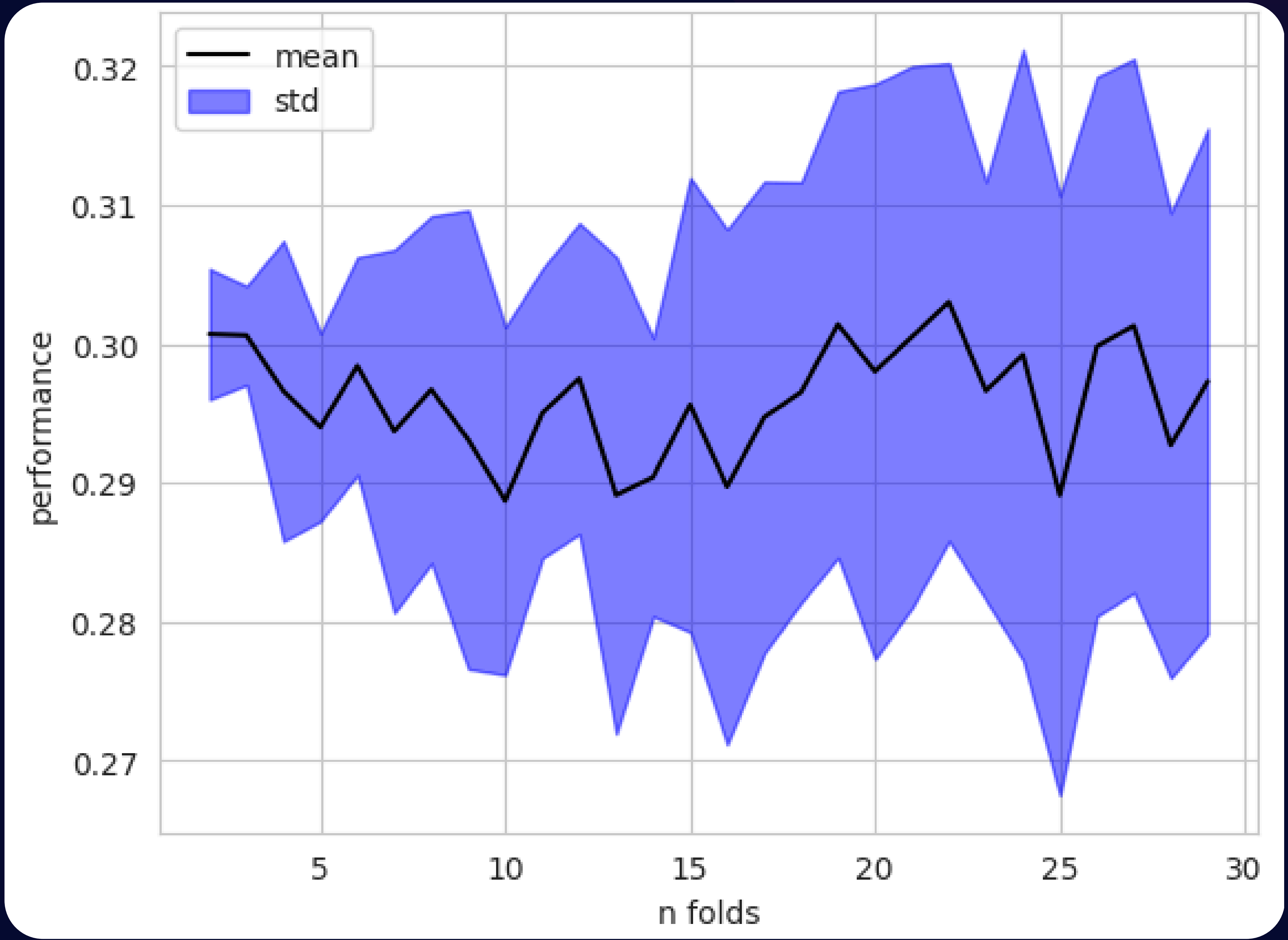
```python
#@title Support Vector Machines
svc = SVC(random_state=42)
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.776
```

N-Folds

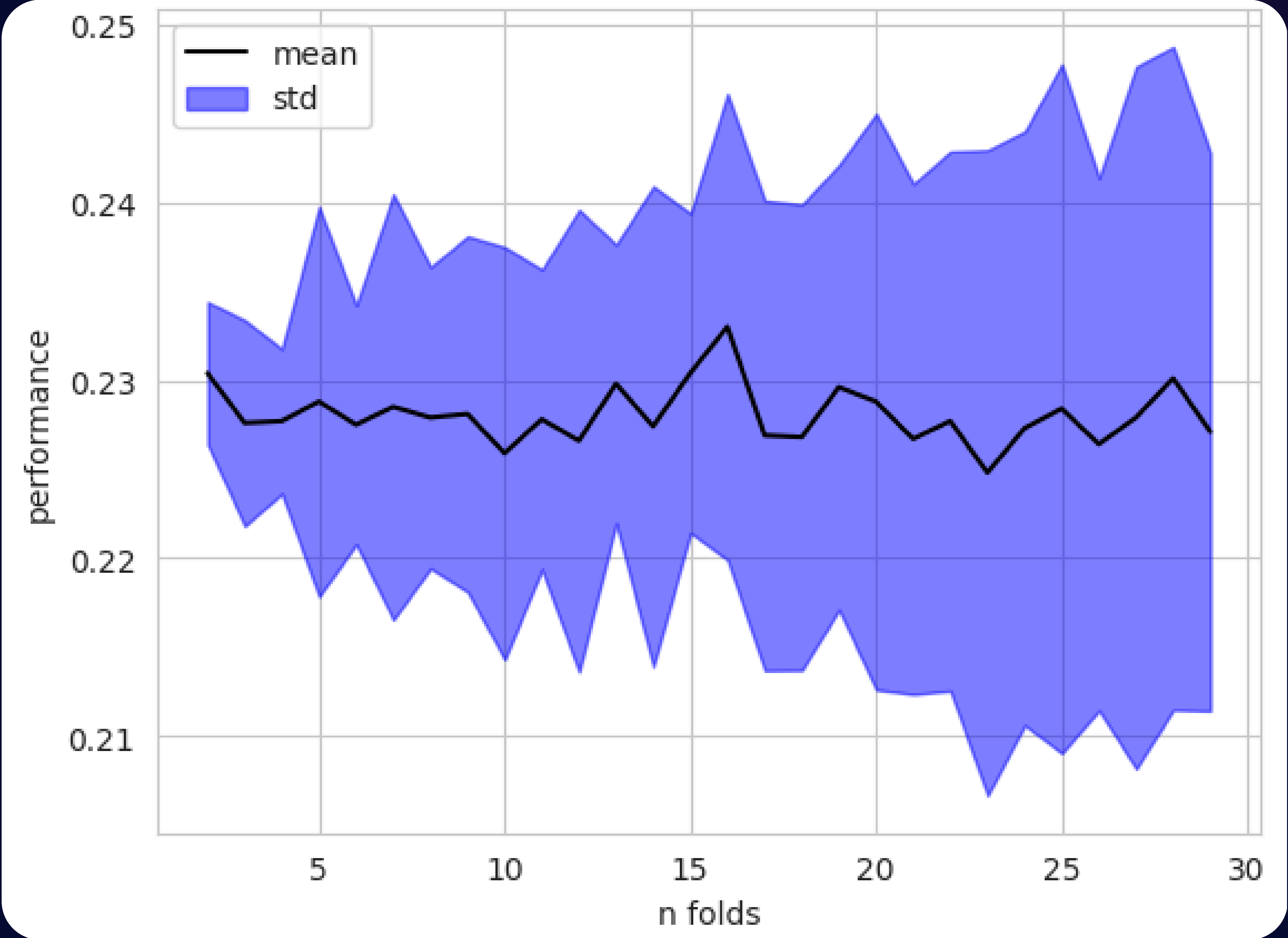
DESICION TREE

N-folds = 10
σ = 0.012522379965485792



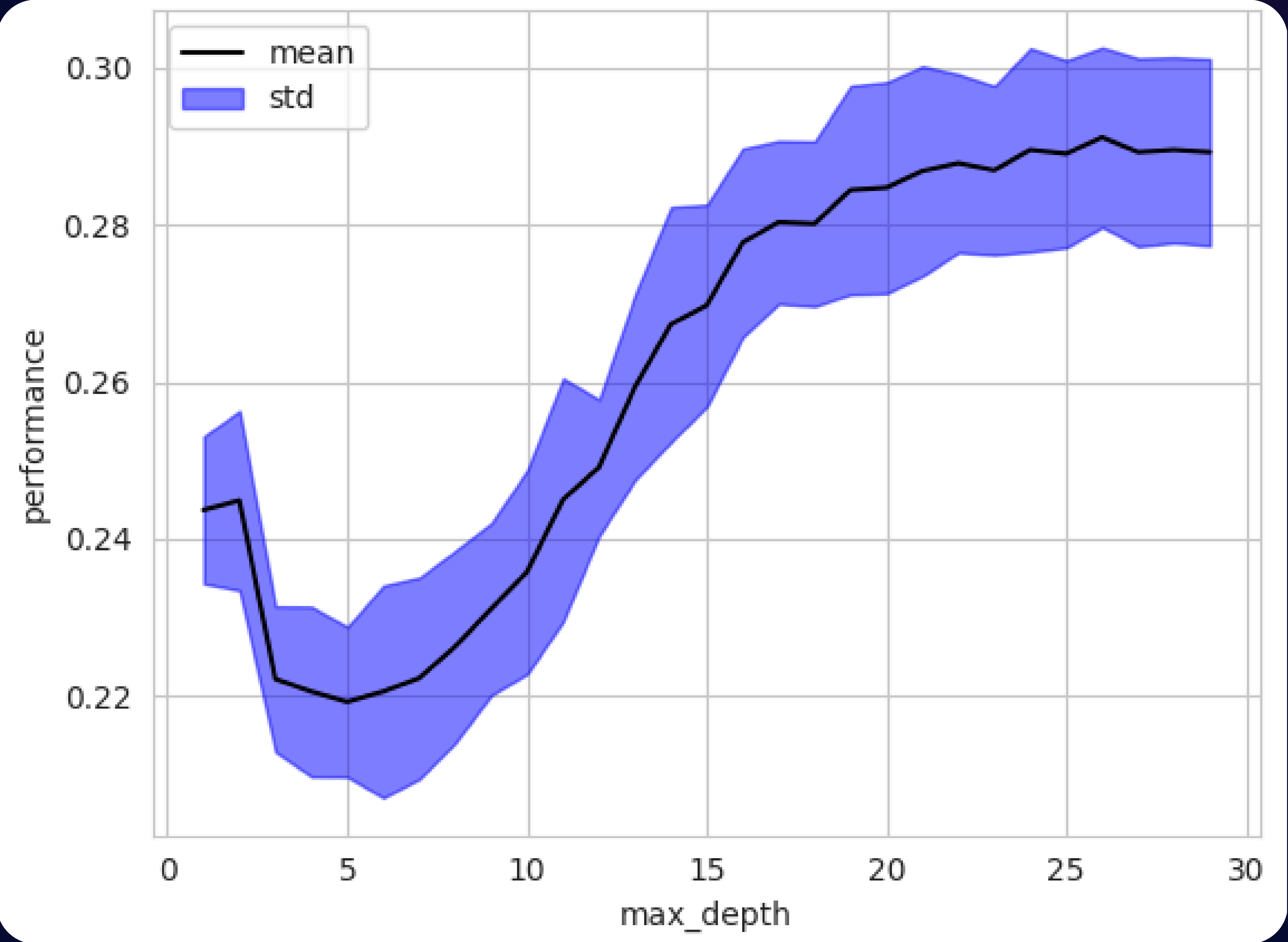
RANDOM FOREST

N-folds = 10
σ = 0.011579723658188051



DESICION TREE

Max depth = 5
σ = 0.009539916142189095

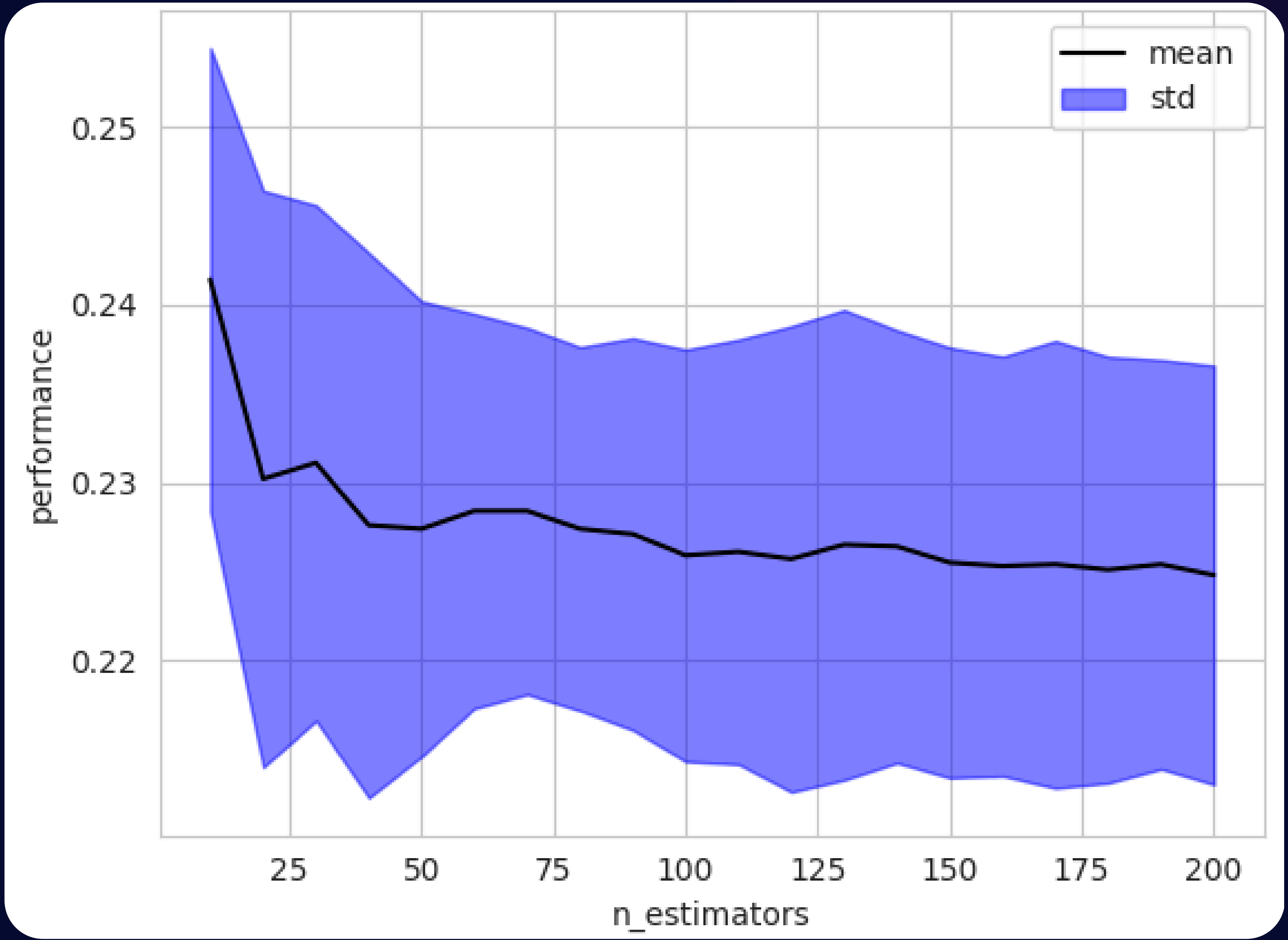


N-Estimators

RANDOM FOREST

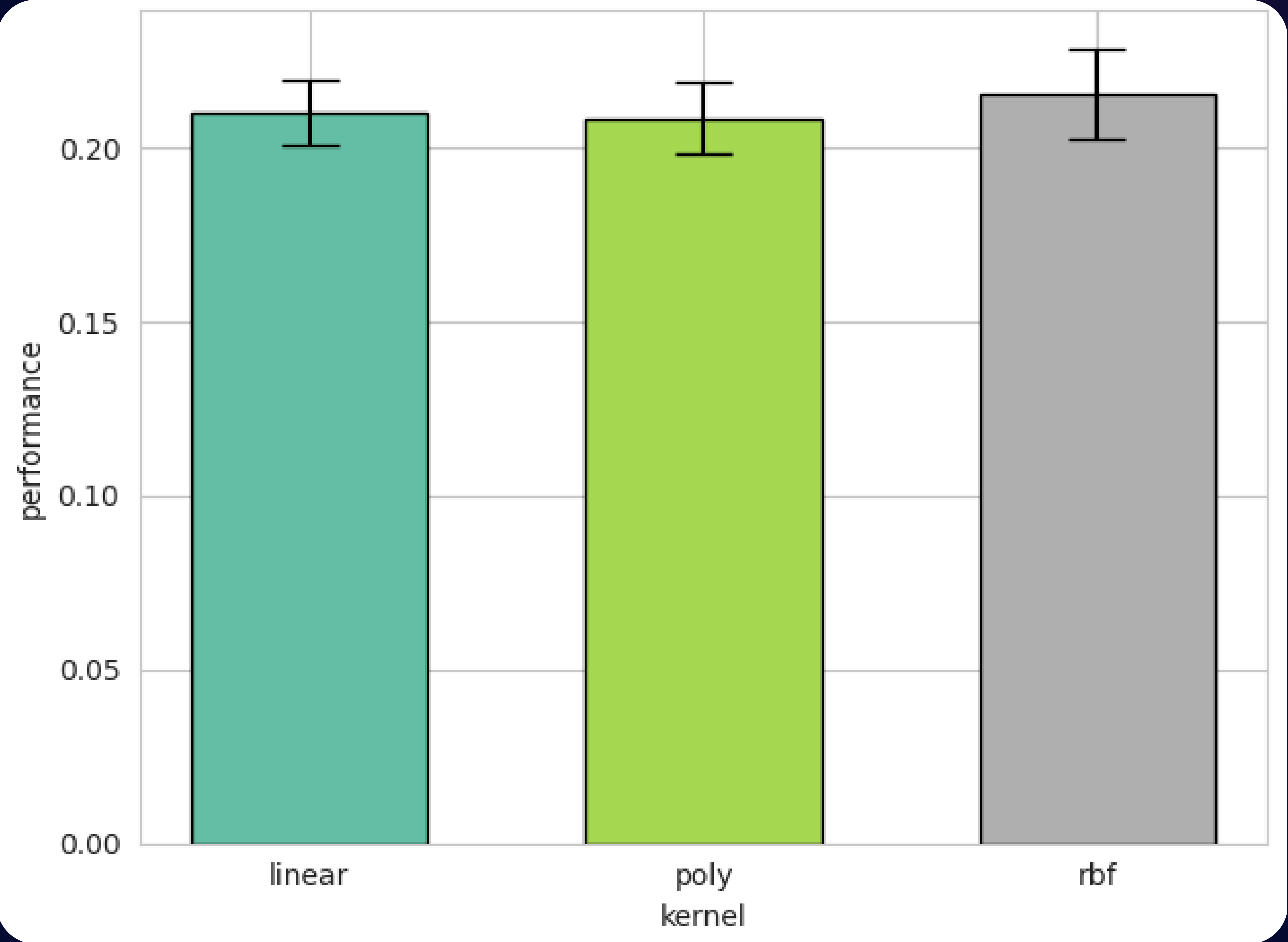
N-Estimators = 170

σ = 0.011762652762026088



SUPPORT VECTOR MACHINE

Kernel = poly

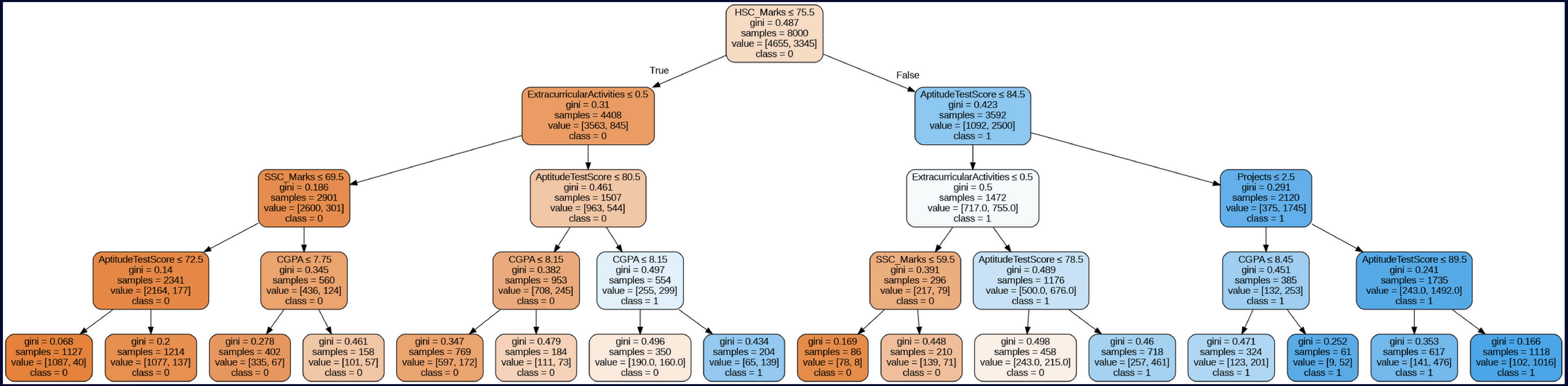


BEST DECISION TREE

```
best_decision_tree = DecisionTreeClassifier(max_depth=5, random_state=42)
best_decision_tree.fit(X_train, y_train)
predictions_tree = best_decision_tree.predict(X_test)
accuracy_tree = accuracy_score(y_test, predictions_tree)
```

```
Accuracy Decision Tree: 0.785
```

TREE DECISIONS FROM BEST DECISION TREE



BEST RANDOM FOREST

```
best_rf = RandomForestClassifier(n_estimators=170, random_state=42)
best_rf.fit(X_train, y_train)
predictions_rf = best_rf.predict(X_test)
accuracy_rf = accuracy
print("Accuracy Random Forest:", accuracy_rf)
```

```
Accuracy Decision Tree: 0.7805
```

BEST SUPPORT VECTOR MACHINE

```
best_svc = SVC(kernel='poly', random_state=42)
best_svc.fit(X_train, y_train)
predictions_svc = best_svc.predict(X_test)
accuracy_svc = accuracy_score(y_test, predictions_svc)
print("Accuracy Support Vector Machine: ", accuracy_svc)
```

```
Accuracy Support Vector Machine:  0.781
```

NEURAL NETWORKS

3 CAPAS OCULTAS, 128 NEURONAS, RELU

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=X_scaled[0].shape),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='sigmoid') #Clasificacion binaria
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy', ])

model.fit(X_scaled, y, epochs=10)
```

```
Test accuracy: 0.8004999756813049
```

6 CAPAS OCULTAS, 128 NEURONAS, RELU

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=X_scaled[0].shape),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='sigmoid') #Clasificacion binaria
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy', ])

model.fit(X_scaled, y, epochs=10)
```

```
Test accuracy: 0.7975000143051147
```

10 CAPAS OCULTAS, 128 NEURONAS, RELU

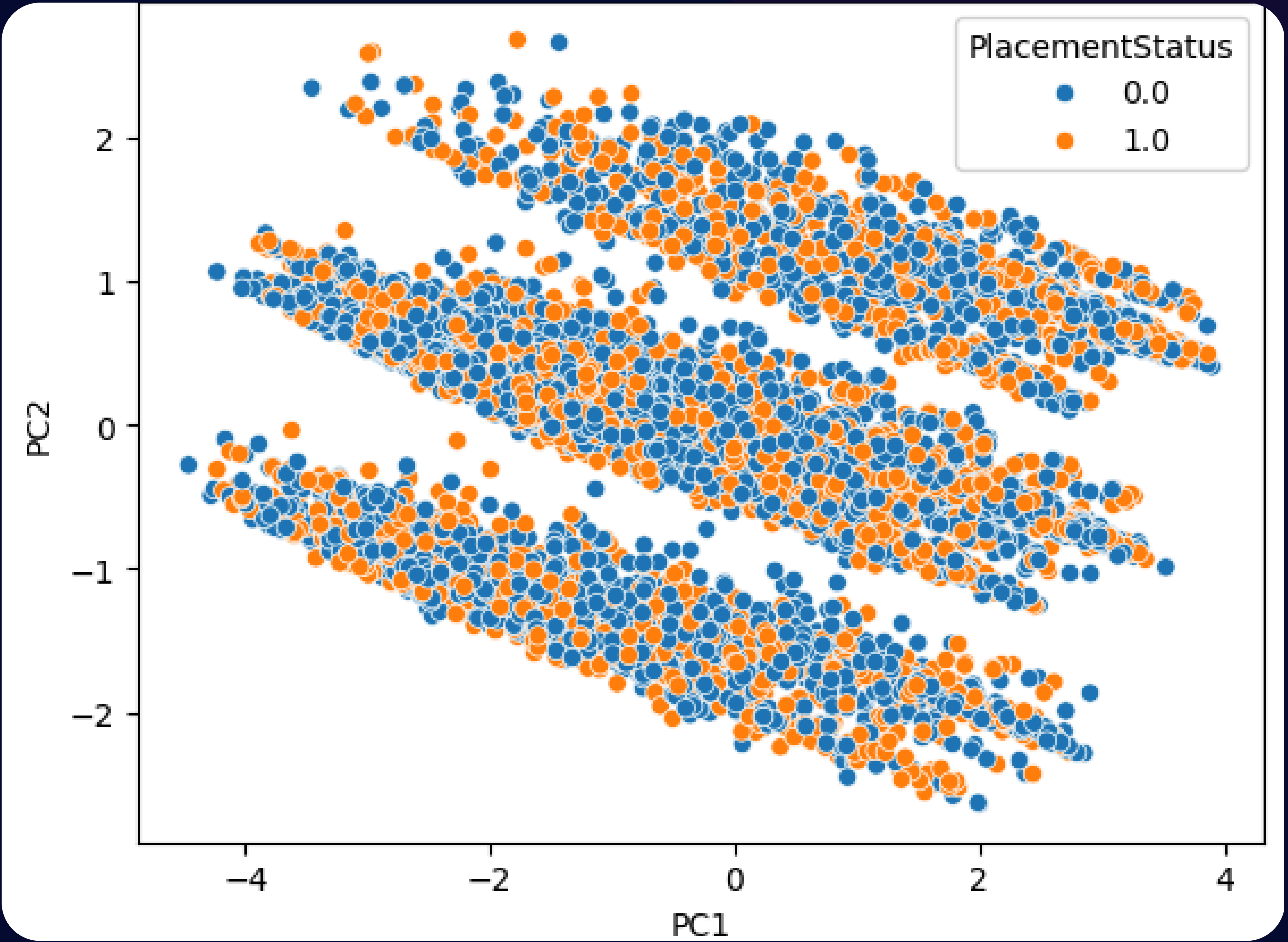
```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=X_scaled[0].shape),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='sigmoid') #Clasificacion binaria
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy', ])

model.fit(X_scaled, y, epochs=10)
```

```
Test accuracy: 0.7914999723434448
```

PRINCIPAL COMPONENT ANALYSIS

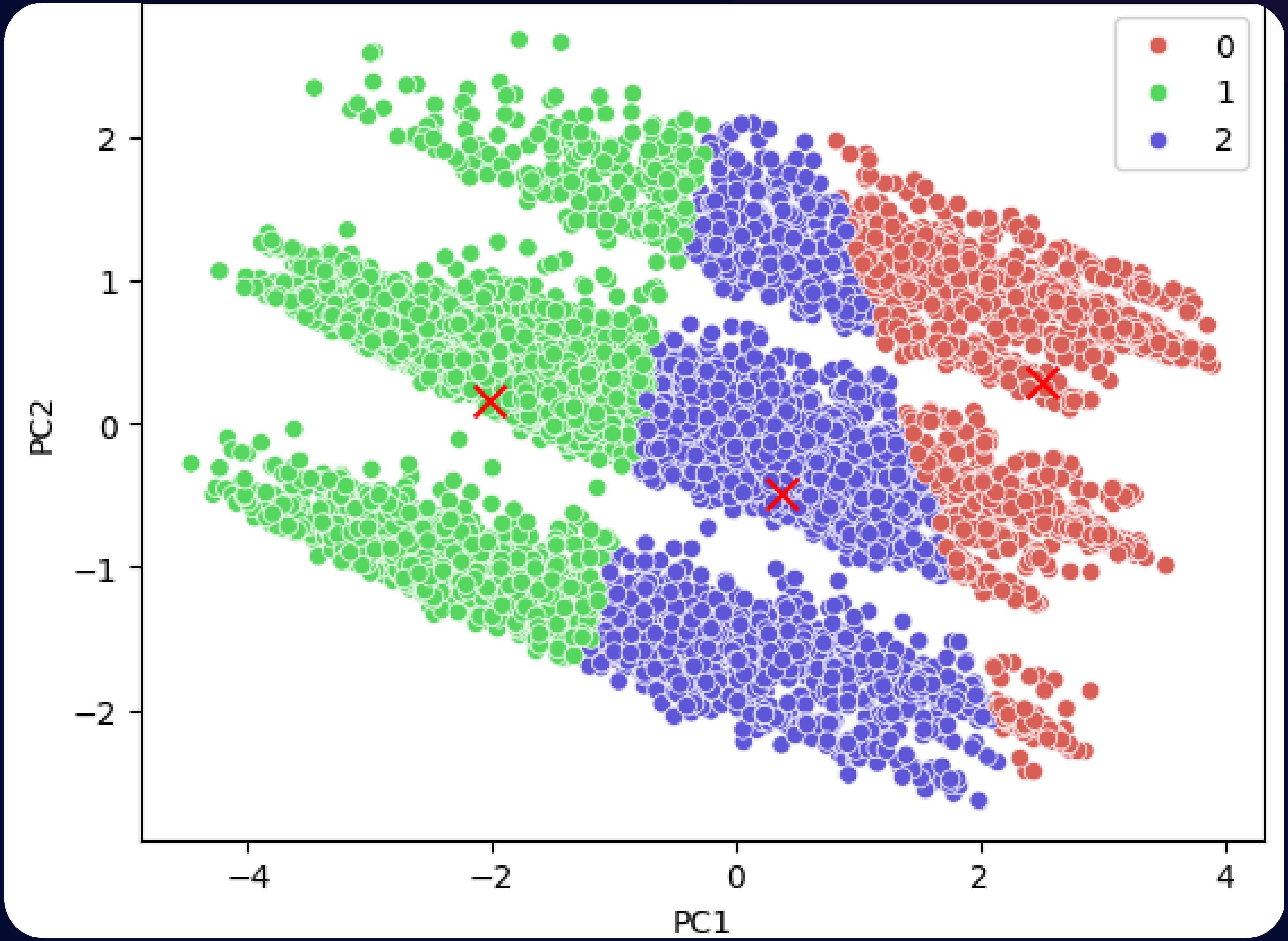
N-Components = 2

La "energia" que conservamos con dos componentes es de 57%



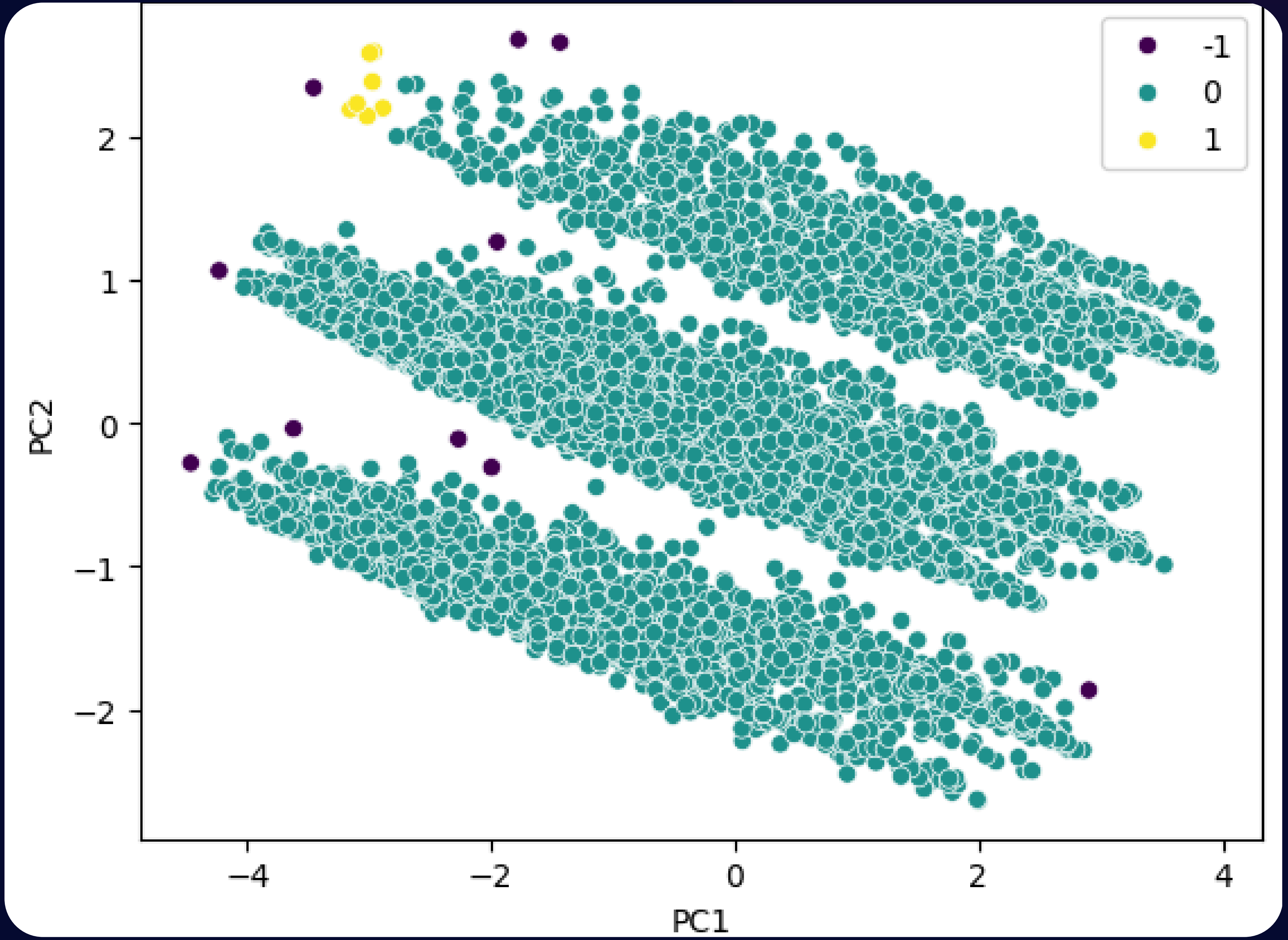
K-MEANS

N-Clusters = 3



DBSCAN

Min-Samples = 2



GRACIAS!