# Contents



## 1 Estructura básica de un modelo con PyCaret

La estructura básica de un proyecto con PyCaret es la del código fuente a continuación:

```
# importar la libreria
from pycaret.[regression/classification] import *

# cargar los datos
data = pd.read_csv("data.csv")

# iniciar una sesión de pycaret
reg = setup(data, target = 'target_column')

# comparar todos los modelos disponibles
compare_models()

# seleccionar el mejor modelo
best = compare_models()[0]

# entrenar el modelo seleccionado
best_model = create_model(best)

# evaluar el modelo en los datos de prueba
evaluate_model(best_model)

# desplegar el modelo
deploy_model(best_model, model_name = 'best_model')
```

## 2 Tarea 2 con PyCaret: Clasificación

- En primer lugar vamos a usar los datos de COVID19 adjuntos a la tarea en la Moodle.
- Cargar el conjunto de datos en el entorno de trabajo. PyCaret admite cargar datos desde un archivo CSV o desde un DataFrame de pandas.
- Inicializar una instancia de PyCaret en el conjunto de datos con la función `setup()`. Especifique la columna objetivo que contiene las etiquetas de clase.

- Utilizando la función `compare_models()`, deben comparar todos los modelos de clasificación disponibles y seleccionar el mejor modelo.
- Una vez que hayan seleccionado el mejor modelo, deben entrenarlo utilizando la función `create_model()`.
- También se puede utilizar `create_model()` para crear un modelo de clasificación automáticamente. PyCaret automáticamente ajustará y seleccionará un conjunto de modelos, como XGBoost, LightGBM, Random Forest, entre otros.
- Utilice la función `tune_model()` para mejorar el rendimiento del modelo seleccionado mediante la búsqueda de mejores hiperparámetros.
- Utilice la función `evaluate_model()` para evaluar el rendimiento del modelo en términos de precisión, recall, F1 y otras métricas.
- Utilice la función `predict_model()` para hacer predicciones en un conjunto de datos de prueba.
- Utilice la función `plot_model()` para visualizar las características más importantes del modelo.
- Finalmente, utilice la función `save_model()` para guardar el modelo entrenado en disco.
- **Opcional** Finalmente, deben desplegar el modelo utilizando la función `deploy_model()`.

### 2.1 Datos y objetivo

Los **datos** a usar son los del **COVID19**,y nuestra **misión** es *decidir* si un **paciente es de alto riesgo** o no.

### 2.2 Preguntas sobre el modelo:

- ¿Cuál es el problema de clasificación que están tratando de resolver?
- ¿Qué modelo de clasificación seleccionaron como el mejor y por qué?
- ¿Cómo evaluaron el rendimiento del modelo en el conjunto de pruebas? ¿Cuáles fueron los resultados?
- ¿Cuáles fueron las métricas de rendimiento que utilizaron para evaluar el modelo?
- ¿Cuáles fueron los hiperparámetros que utilizaron para mejorar el rendimiento del modelo?
- ¿Cómo visualizaron los resultados del modelo? ¿Qué información pueden extraer de la visualización?
- ¿Cómo interpretaron el modelo? ¿Qué características o variable son más relevantes en la toma de decisiones del modelo?
- ¿Cómo desplegaron el modelo y cómo podría ser utilizado en una aplicación real?

Es importante consultar la documentación oficial

## 2.3 Ejemplo

Este ejemplo utiliza el conjunto de datos de iris incluido en PyCaret. El conjunto de datos de iris tiene cuatro características (largo y ancho del sépalo y pétalo) y tres especies diferentes (setosa, versicolor y virginica). El objetivo es construir un modelo de clasificación que pueda predecir la especie a la que pertenece una flor de iris basándose en sus características.

Ten en cuenta que para el uso de la funcion create_model y tune_model es recomendable tener instalado Xgboost, Catboost y/o LightGBM.

```
# importar las librerias necesarias
from pycaret.datasets import get_data
from pycaret.classification import *

# cargar el dataset de iris
data = get_data('iris')

# inicializar PyCaret y especificar la columna objetivo
clf1 = setup(data, target = 'species')

# crear un modelo de clasificación automáticamente
best_model = create_model('xgboost')

# mejorar el rendimiento del modelo
tuned_model = tune_model(best_model)

# evaluar el rendimiento del modelo
evaluate_model(tuned_model)

# hacer predicciones en un conjunto de datos de prueba
predictions = predict_model(tuned_model, data=data)

# visualizar las características más importantes del modelo
plot_model(tuned_model, plot = 'feature')

# guardar el modelo entrenado en disco
save_model(tuned_model, 'tuned_xgboost_model')
```

### 2.3.1 Más ejemplos

Ejemplo más completo 1 en GitHub, introductorio

Ejemplo más completo 2 en GitHub, intermedio