PROYECTO MACHINE LEARNING

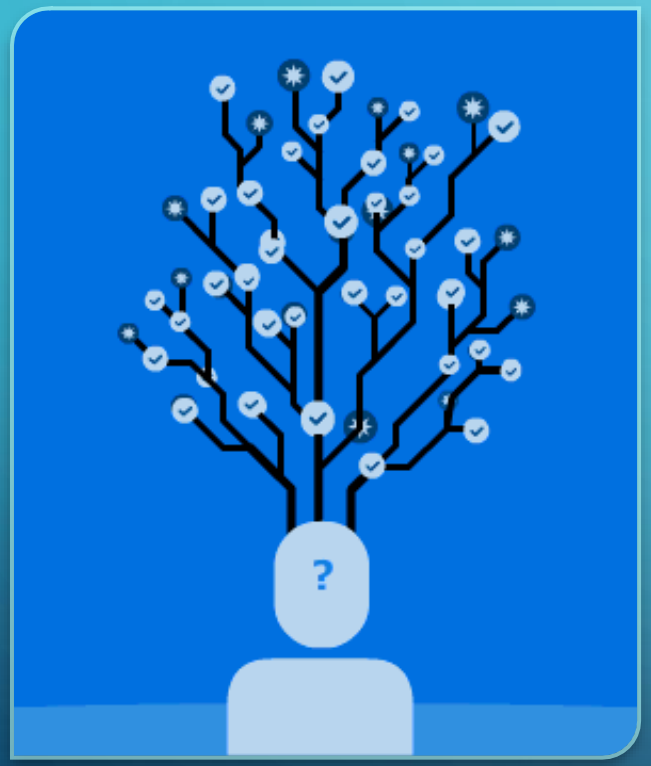
PREDICCIONES SPEED DATING

BY DARÍO RUIZ

1. PROBLEMA DE NEGOCIO

- Organizar un evento de citas rápidas (*speed dating)*
- Datos obtenidos de un estudio realizado por un tercero
- Mayor cantidad de *matches* posible
- Aspectos a tener en cuenta

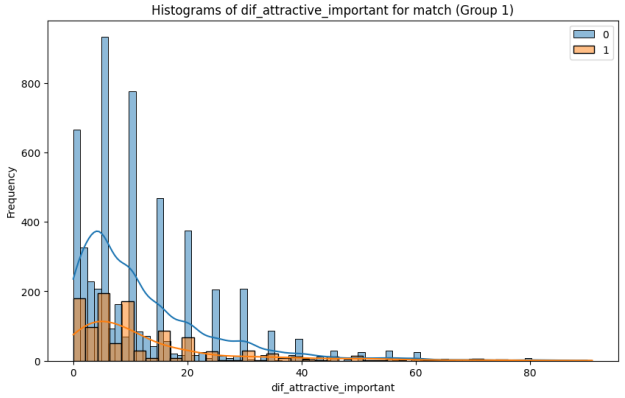
2. PROBLEMA DE MACHINE LEARNING

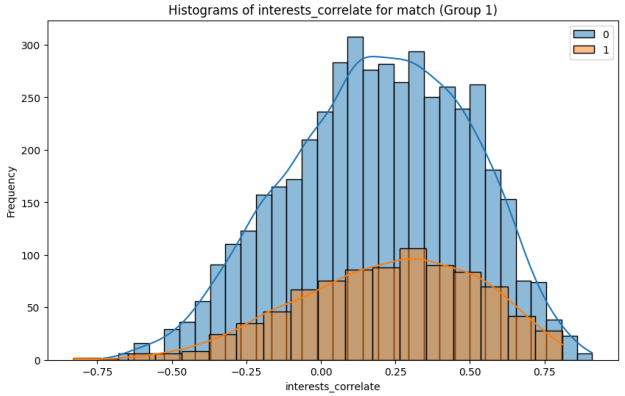


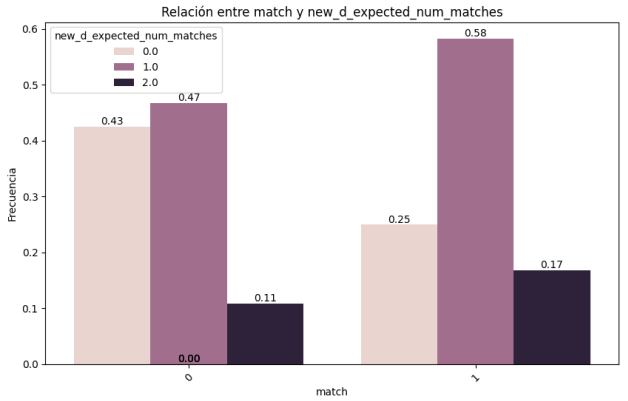
- Problema de clasificación binaria
- Métrica: sensibilidad media *(balanced accuracy)*
- Modelo que permita obtener la importancia de las features: indicar a negocio cómo debe hacer los emparejamientos

3. DATASET

- Datos obtenidos obtenidos por Columbia Business School en varios eventos de *speed dating*
- Entre 2002 y 2004, algunos sesgos, solo parejas heterosexuales
- Cada instancia es una cita entre dos participantes: variables demográficas, intereses y preferencias de una pareja







4. EDA

- Target desbalanceado : 84% (No) vs. 16% (Sí)
- Algunas variables:
  - Correlación de los intereses
  - Diferencia entre la importancia del atractivo
  - Número de *matches* esperado (categórica)

5. MODELOS Y OPTIMIZACIÓN

```python
def objective(trial):

    param_grid = {
    "n_estimators": trial.suggest_int("n_estimators", 100, 600, step = 10),
    "max_depth": trial.suggest_int("max_depth", 3, 20),
    "learning_rate": trial.suggest_float("learning_rate", 0.01, 1, step = 0.01),
    "num_leaves": trial.suggest_int("num_leaves", 15, 150, step = 5),
    "colsample_bytree": trial.suggest_float("colsample_bytree", 0.5, 1.0, step = 0.1),
    "min_child_samples": trial.suggest_int("min_child_samples", 1, 800, step = 10),
    "subsample": trial.suggest_float("subsample", 0.5, 1, step = 0.1),
    "class_weight": trial.suggest_categorical("class_weight", ["balanced", None]),
    "random_state": 42,
    "verbose": -100
    }

    model = LGBMClassifier(**param_grid)
    model.fit(X_train,y_train)

    cv = KFold(n_splits= 6, shuffle= True, random_state= 42)

    cv = cross_val_score(model, X_train, y_train, cv=cv, scoring= "balanced_accuracy").mean()
    return cv

study_lgbmc = optuna.create_study(direction= "maximize")
study_lgbmc.optimize(objective, n_trials=300)
```

- Baseline: 62% de *balanced accuracy* en CV
- Optimización: XGBoost, KNN, LightGBM
- Mejor modelo: LightGBM, 82% en CV
- Generaliza bien: 86% en test

5. CONCLUSIONES Y ACCIONES DE MEJORA

- Modelo robusto y explicable: *4 features*
- *Feature importance:* cómo hacer las parejas
- Asumir compromiso: mejorar el *recall*, incluso a costa de reducir la precisión en cierta medida

¡MUCHAS GRACIAS!