

Proyecto final Data Mining

ANÁLISIS DE LA EMPRESA SHOWZ

Realizado por: Melanie Álvarez, Marie Cucalón, Fabián
De La Cruz, Roberth Lara, Ma. Emilia Rivadeneira



Entendimiento del Negocio

El éxito ya no se mide solo por las ventas inmediatas, sino por el valor a largo plazo que cada cliente aporta (LTV) y el costo asociado a su adquisición (CAC). Las fuentes de adquisición (como campañas digitales u orgánicas) presentan distintas eficiencias y niveles de retención.

Construir un sistema de predicción basado en inteligencia artificial que permita:

- Anticipar el LTV esperado por cliente.
- Calcular el CAC por fuente y cohorte.
- Identificar los drivers clave que influyen en estas métricas.
- Simular escenarios para optimizar el presupuesto de marketing y maximizar el ROMI.

Entendimiento de los Datos

02

Se trabajó con tres archivos clave que representan distintas dimensiones del negocio:

01

visits

+350K sesiones. Predominio de dispositivos desktop, 10 fuentes de adquisición, y cálculo de duración de sesión. Se detectaron 24,658 outliers conservados por su posible valor informativo.

02

orders

Detalles de compras con estacionalidad alta en Q4. 3,990 outliers en ingresos retenidos para modelar usuarios de alto valor.

03

costs

Gastos en marketing crecientes hacia fin de año. Se hallaron 189 outliers asociados a campañas intensas, conservados por su relevancia en el CAC.



Preparación de datos

01

Variables de comportamiento

Permiten capturar patrones de lealtad, valor e intensidad de compra, fundamentales para predecir el LTV.

Ejemplos: aov, revenue, dias_entre_compras, frecuencia_compras, dias_primera_sesion_a_primera_compra

02

Variables temporales

Ayudan a capturar efectos de tiempo que puedan influir en el valor o costo de adquisición del cliente.

Ejemplos: mes, dia de la semana, estación, es_fin_de_semana (de la primera_sesion), dias_activo, cohort_year y cohort_month

03

Variables de marketing

Importantes para modelar la conversión de marketing en valor (ROMI) y los costos asociados.

Ejemplos: primer_source y primer_dispositivo, n_sesiones, n_dispositivos_distintos:, duracion_promedio_sesion:

04

Etiquetas

LTV_180: suma del ingreso generado por un usuario en los 180 días posteriores a su primera sesión.

CAC_source_30: costo promedio de adquisición por fuente.



LTV

El modelo Random Forest logró una alta precisión ($MAE = \$1.36$, $MAPE = 2.17\%$), lo cual permite a Showz estimar con confianza el ingreso futuro por cliente y asignar presupuestos personalizados.

CAC

Todos los modelos tuvieron errores relativos altos ($MAPE > 40\%$), superando incluso por poco a una predicción basada en la mediana. Esto se debe a la falta de granularidad en los datos de marketing y a que el CAC es muy bajo, haciendo que cualquier error tenga alto impacto porcentual.

Modelamiento

- Una función para cada modelo
- Una función de entrenamiento y evaluación global
- Una función que divide el dataset
- Una ‘función maestra’

```
#regresion Lineal
def modelo_lineal(*args):
    return modelo_con_gridsearch(LinearRegression(), {}, *args)

#estocastico
def modelo_estocastico(*args):
    param_grid = {'alpha': [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100], 'penalty': ['l2', 'elasticnet']}
    return modelo_con_gridsearch(SGDRegressor(max_iter=1000, tol=1e-3, random_state=RANDOM_STATE), param_grid, *args)

#ridge
def modelo_ridge(*args):
    param_grid = {'alpha': [0.1, 1.0, 10.0]}
    return modelo_con_gridsearch(Ridge(), param_grid, *args)
```

```
# División del dataset -> 6 meses de 2017 y 2 trimestres de 2018
def dividir_dataset(df, fecha_col='fecha_primera_sesion'):
    df[fecha_col] = pd.to_datetime(df[fecha_col])
    train = df[df[fecha_col].dt.year == 2017]
    val = df[(df[fecha_col].dt.year == 2018) & (df[fecha_col].dt.month <= 3)]
    test = df[(df[fecha_col].dt.year == 2018) & (df[fecha_col].dt.month > 3)]
    return train, val, test
```



```

def modelo_con_gridsearch(modelo_base, param_grid,X_train, X_val, X_test, y_train, y_val, y_test, numeric_cols=None):
    if numeric_cols is None:
        numeric_cols = X_train.columns
    preproc = ColumnTransformer([
        ("scale", StandardScaler(), numeric_cols)],
        remainder="drop"
    ).set_output(transform="pandas")

    pipeline = Pipeline([
        ("pre", preproc),
        ("modelo", modelo_base)
    ])

    param_grid_pipeline = {f"modelo_{k}": v for k, v in param_grid.items()}

    tscv = TimeSeriesSplit(n_splits=5)
    grid = GridSearchCV(
        pipeline,
        param_grid_pipeline,
        cv=tscv,
        scoring="neg_root_mean_squared_error",
        refit=True
    )
    grid.fit(X_train, y_train)
    print("Mejores parámetros:", grid.best_params_)
    best_model = grid.best_estimator_

    # evaluación en validación
    y_pred_val = best_model.predict(to_df(X_val, X_train))
    val_scores = metricas(y_val, y_pred_val)
    print(f"\n[VALIDACIÓN] MAE={val_scores['MAE']:.2f}, "
          f"RMSE={val_scores['RMSE']:.2f}, MAPE={val_scores['MAPE']:.2f}%")

    # refit en train+val
    X_train_val = pd.concat([X_train, X_val], axis=0)
    y_train_val = pd.concat([y_train, y_val], axis=0)
    best_model.fit(X_train_val, y_train_val)

    # evaluación final en test
    y_pred_test = best_model.predict(to_df(X_test, X_train))
    test_scores = metricas(y_test, y_pred_test)
    print(f"\n[TEST] MAE={test_scores['MAE']:.2f}, "
          f"RMSE={test_scores['RMSE']:.2f}, MAPE={test_scores['MAPE']:.2f}%")

    return best_model, {"val": val_scores, "test": test_scores}

```

```

def ejecutar_modelos(df, target_col, fecha_col='fecha_primera_sesion'):
    # Definimos las columnas a excluir del modelo
    columnas_excluir = [
        target_col,
        fecha_col,
        'uid',
        'fecha_primera_compra',
        'fecha_ultima_compra',
        'fecha_primera_sesion',
        'primer_source'
    ]

    # Limpiamos nulos
    df = df.dropna(subset=[target_col]).dropna()

    # Particionamos
    train, val, test = dividir_dataset(df, fecha_col)

    # X / y
    X_train, y_train = train.drop(columns=columnas_excluir), train[target_col]
    X_val, y_val = val.drop(columns=columnas_excluir), val[target_col]
    X_test, y_test = test.drop(columns=columnas_excluir), test[target_col]

    modelos = {}
    evaluaciones = {}

    print("\nModelos de regresión:")
    print("\nLineal\n")
    modelos['lineal'], evaluaciones['lineal'] = modelo_lineal(X_train, X_val, X_test, y_train, y_val, y_test)
    print("\nEstocástico\n")
    modelos['estocastico'], evaluaciones['estocastico'] = modelo_estocastico(X_train, X_val, X_test, y_train, y_val, y_test)
    print("\nRidge\n")
    modelos['ridge'], evaluaciones['ridge'] = modelo_ridge(X_train, X_val, X_test, y_train, y_val, y_test)

    print("\n\nModelos avanzados:")
    print("\nRandom Forest\n")
    modelos['rf'], evaluaciones['rf'] = modelo_rf(X_train, X_val, X_test, y_train, y_val, y_test)
    print("\nXGBoost\n")
    modelos['xgb'], evaluaciones['xgb'] = modelo_xgb(X_train, X_val, X_test, y_train, y_val, y_test)
    print("\nLightGBM\n")
    modelos['lgbm'], evaluaciones['lgbm'] = modelo_lgbm(X_train, X_val, X_test, y_train, y_val, y_test)
    print("\nCatBoost\n")
    modelos['catboost'], evaluaciones['catboost'] = modelo_catboost(X_train, X_val, X_test, y_train, y_val, y_test)

    print("\n\nModelos ensambladores:")
    print("\nStacking\n")
    modelos['stacking'], evaluaciones['stacking'] = modelo_stacking(X_train, X_val, X_test, y_train, y_val, y_test)
    print("\nBlending\n")
    modelos['blending'], evaluaciones['blending'] = modelo_blending(X_train, X_val, X_test, y_train, y_val, y_test)

```



LTV / Blending		CAC / Ridge		LTV_180 CAC_30	MAE	MAPE	RMSE
MAE	0.64	MAE	0.13	Mean Median	5.56 3.52	430.84% 155.10%	13.81 13.68
MAPE	5.11%	MAPE	42.43%	Mean Median	0.13 0.11	42.56% 28.29%	0.15 0.17
RMSE	3.26	RMSE	0.15				

primer_source	avg_LTV	avg_CAC	n_users	ROMI	budget
1	9.796521	0.330681	2899	29.625266	958.645055
2	13.000894	0.329137	3506	39.499962	1153.953870
3	5.984117	0.331510	10473	18.051077	3471.906927
4	5.898427	0.330774	10296	17.832208	3405.646750
5	7.917749	0.331241	6931	23.903262	2295.833821
7	2.150563	0.343806	1	6.255168	0.343806
9	5.306426	0.325173	1088	16.318796	353.787873
10	4.162009	0.331131	1329	12.569061	440.073437

source	base_rev	rev_+10%_solo	delta_+10%	rev_redistrib	delta_redistrib
3	263567.87993	269835.045883	6267.165953	271142.951443	7575.071513
4	263567.87993	269640.900036	6073.020106	270998.383750	7430.503820
5	263567.87993	269055.671746	5487.791815	268576.973129	5009.093199
2	263567.87993	268125.993354	4558.113423	266085.598081	2517.718151
1	263567.87993	266407.891402	2840.011472	265659.469654	2091.589724
9	263567.87993	264145.219128	577.339198	264339.780934	771.901004
10	263567.87993	264121.010898	553.130968	264528.040392	960.160462
7	263567.87993	263568.094986	0.215056	263568.630052	0.750122

ya no quiero :(



Fuente 3

Fuente 3

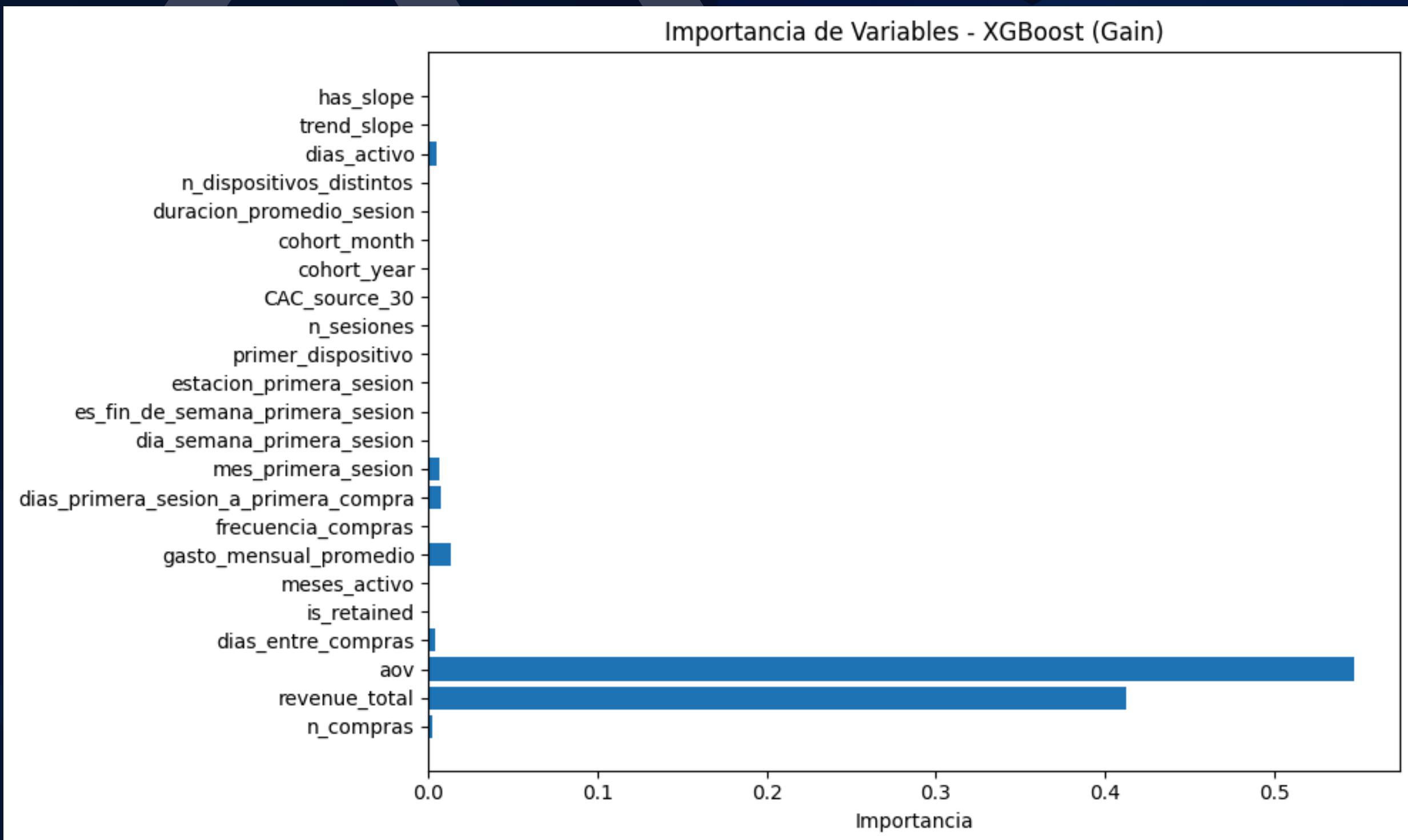
Fuente 3



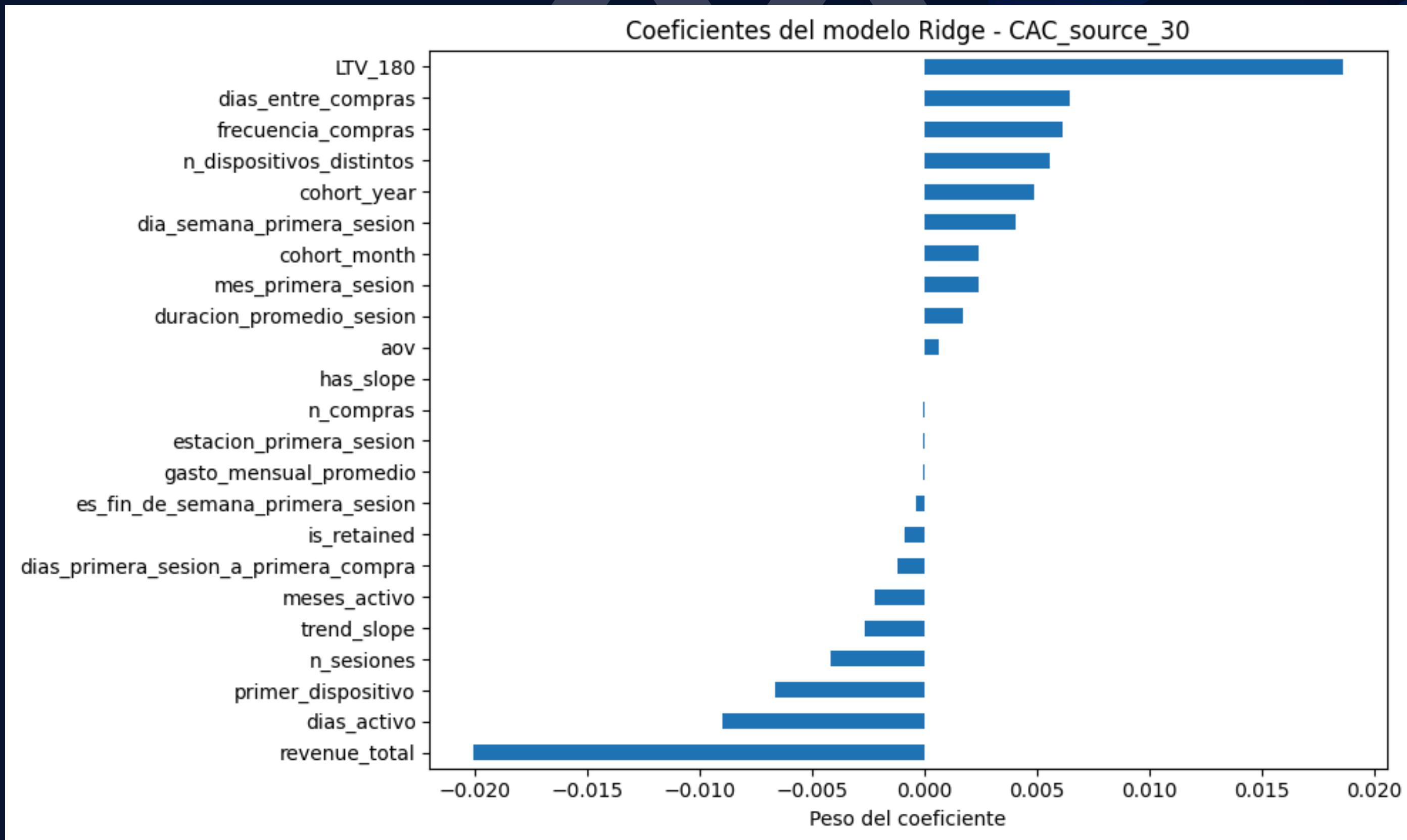
Explicabilidad y Diagnóstico



Importancia de variables - LTV

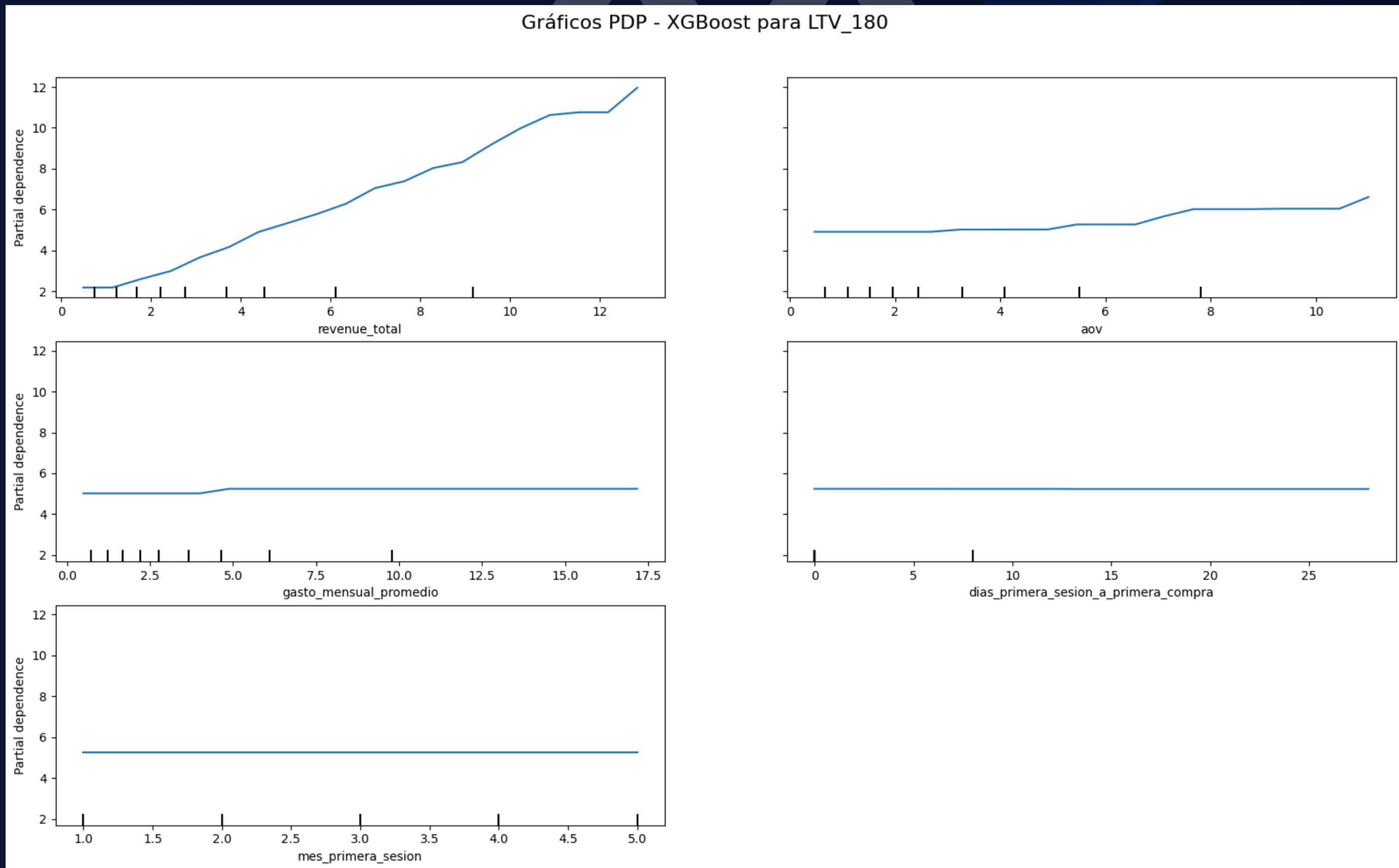


Importancia de variables - CAC



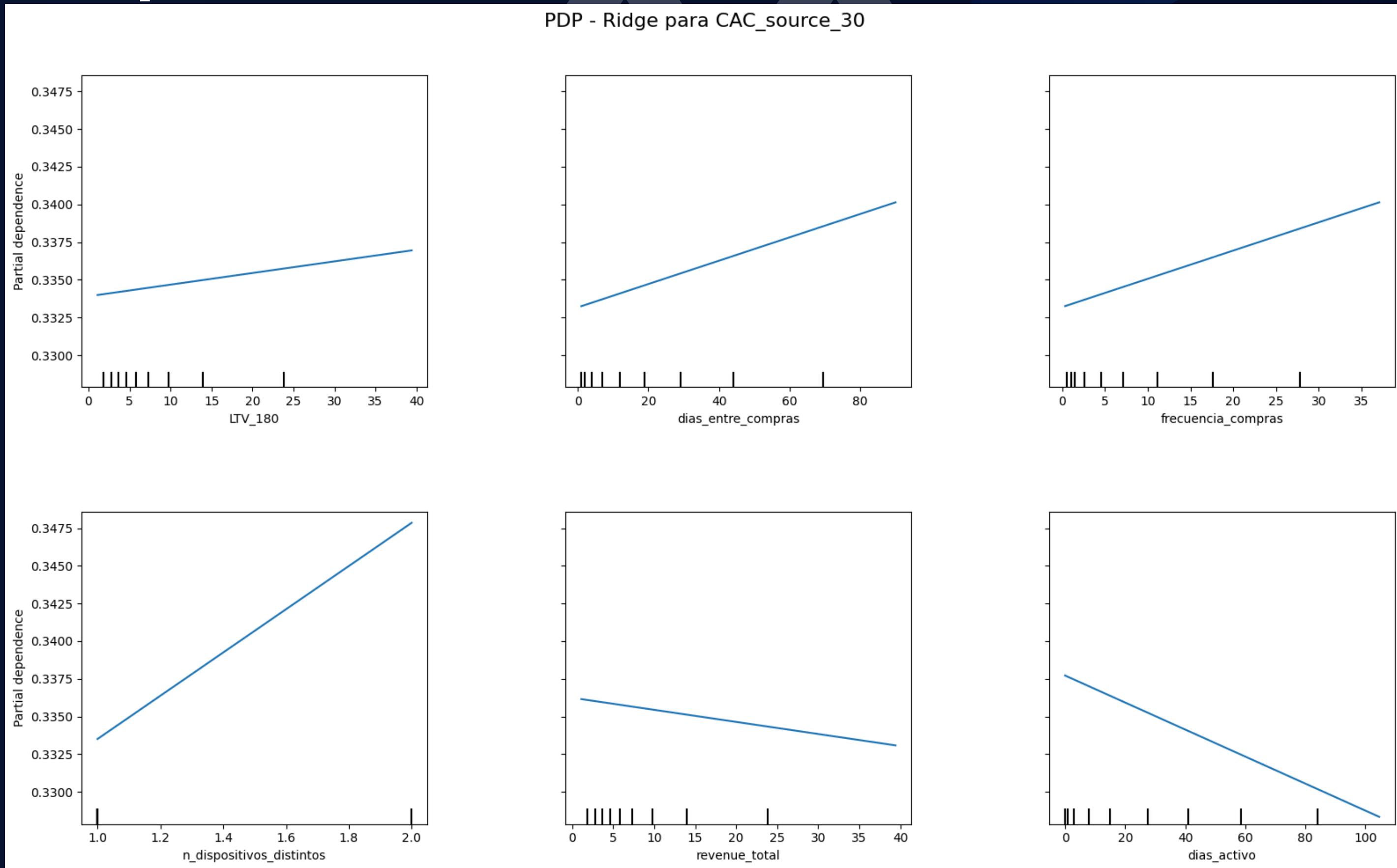
Partial Dependence Plot - LTV

Gráficos PDP - XGBoost para LTV_180

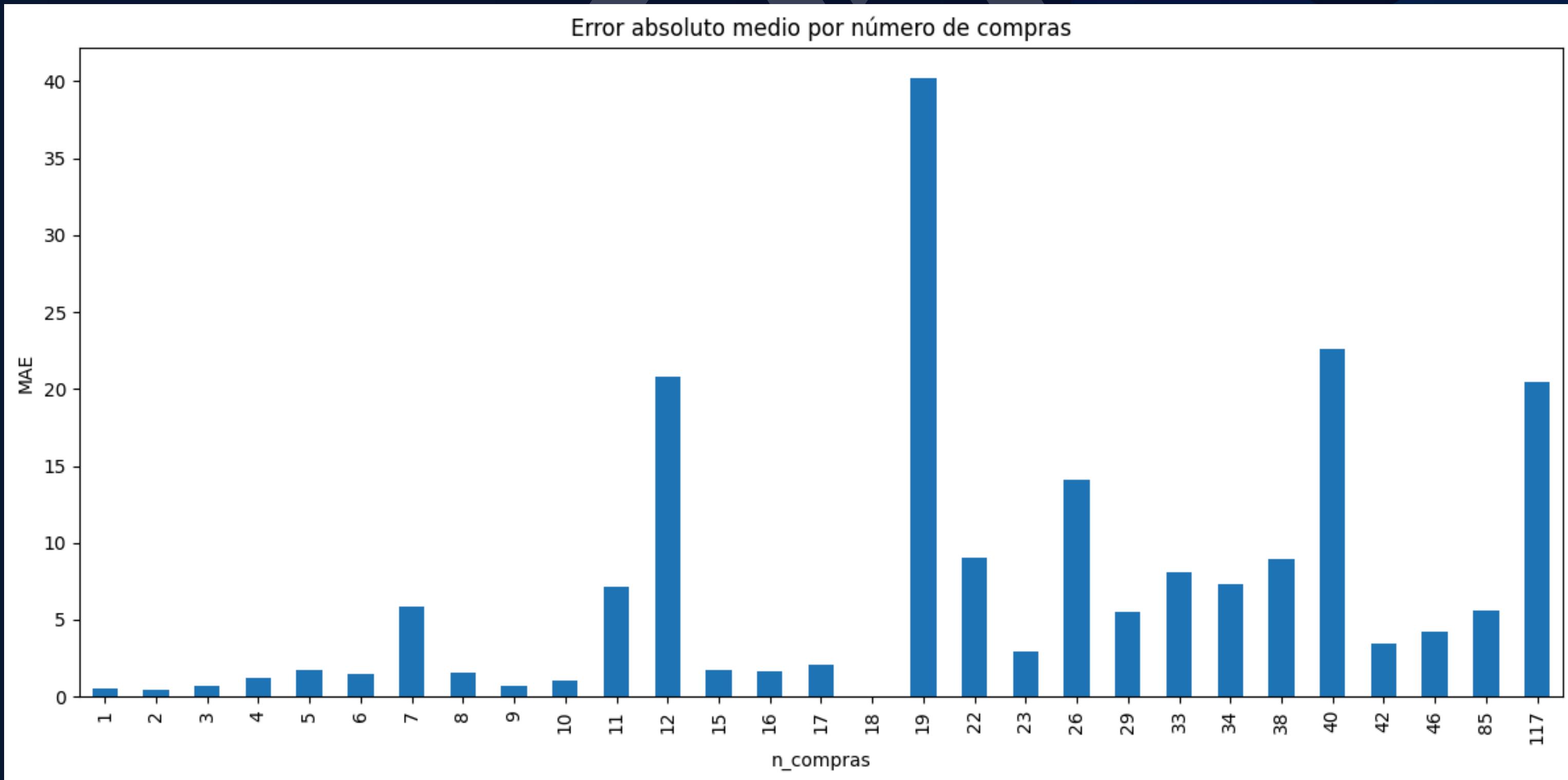


Partial Dependence Plot - CAC

PDP - Ridge para CAC_source_30



Análisis de errores sistemáticos - LTV



Análisis de errores sistemáticos - LTV

