



# PREDICCIÓN DE PRECIOS Y DETECCIÓN DE OPORTUNIDADES EN AIRBNB MADRID

*Sandra López*

DATA SCIENCE BOOTCAMP, THE BRIDGE



# PROBLEMA DE NEGOCIO

*¿Qué queremos resolver?*

OBJETIVO: PREDECIR EL PRECIO ADECUADO DE UN ALOJAMIENTO Y DETECTAR LISTADOS POR DEBAJO DEL VALOR ESTIMADO.



- PROPIETARIOS NECESITAN FIJAR PRECIOS COMPETITIVOS.
- INVERSORES QUIEREN IDENTIFICAR APARTAMENTOS BARATOS RESPECTO AL MERCADO.

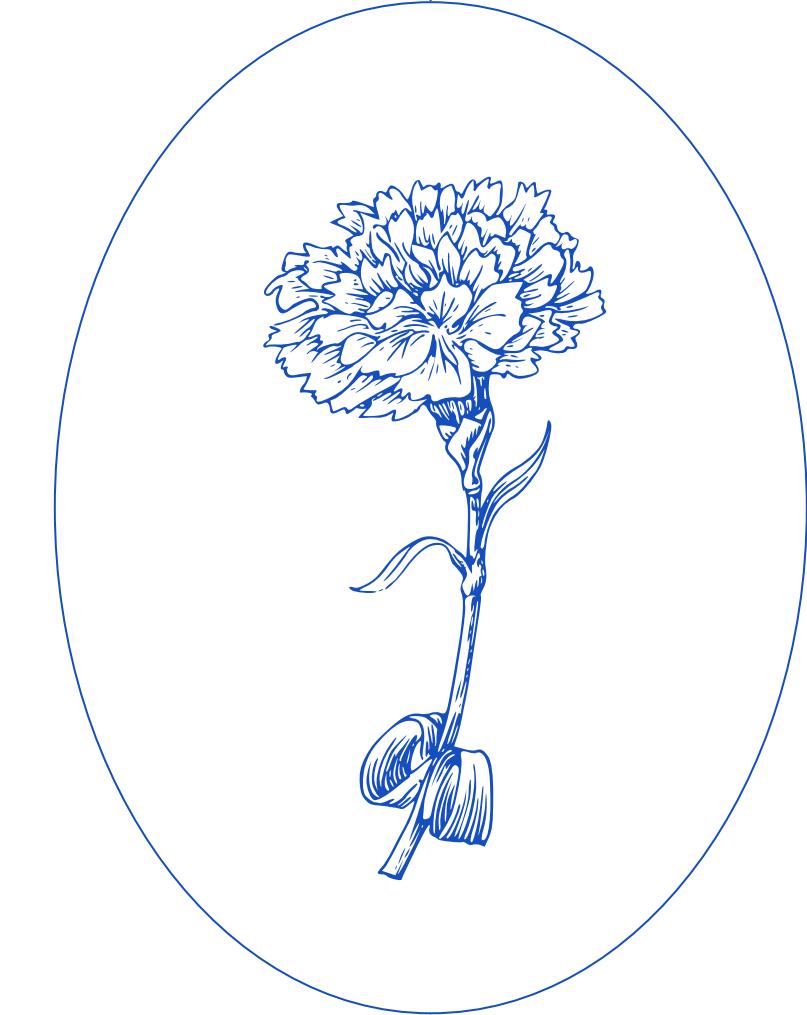
# PROBLEMA TÉCNICO

- TIPO DE PROBLEMA:  
REGRESIÓN SUPERVISADA.
- TARGET: VARIABLE PRICE.

**¿CÓMO AYUDA EL MODELO? →  
ESTIMA EL PRECIO ESPERADO  
SEGÚN ZONA, TIPO,  
DISPONIBILIDAD, ETC., Y LO  
COMPARA CON EL PRECIO REAL  
PARA DETECTAR  
INFRAVALORACIONES.**

# FUENTE

- FUENTE: INSIDE AIRBNB  
(MADRID)
- N° REGISTROS TRAS  
LIMPIEZA: ~19.000
- VARIABLES MÁS  
RELEVANTES: ROOM\_TYPE,  
NEIGHBOURHOOD,  
AVAILABILITY\_365,  
MINIMUM\_NIGHTS, ETC.
- MINIEDA: DISTRIBUCIÓN DEL  
PRECIO (SESGO), OUTLIERS,  
ZONAS MÁS CARAS





```
Def plot_target_distribution_continuous(df, target,  
bins=30, kde=True, color="skyblue"):  
    """
```

Grafica la distribución de una variable continua con histograma y KDE opcional.

Args:

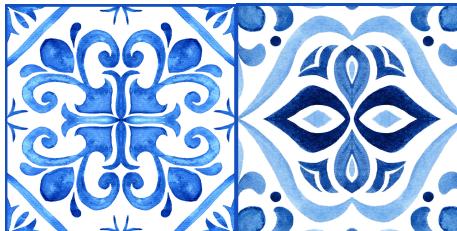
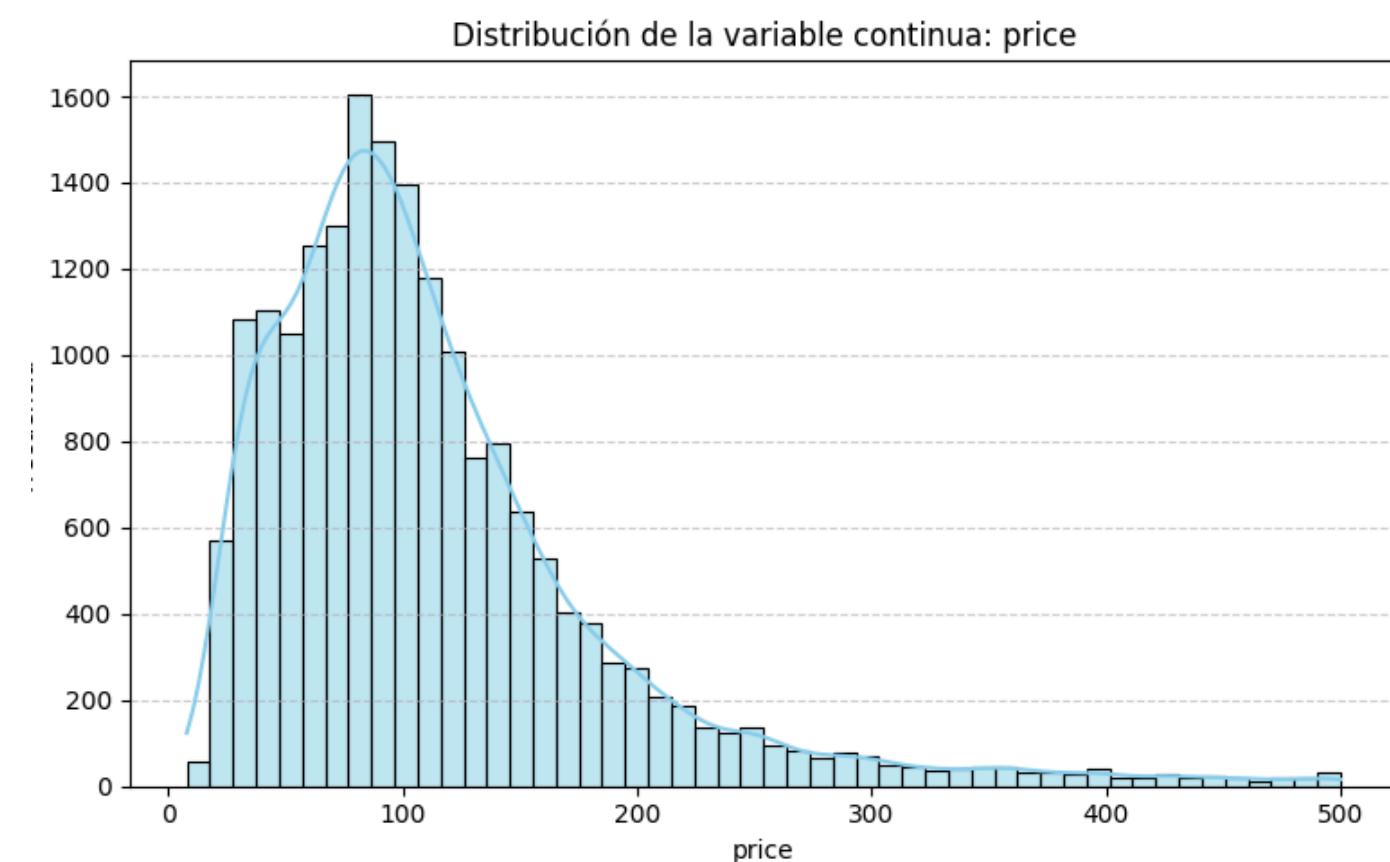
df (pd.DataFrame): DataFrame que contiene los datos.  
target (str): Nombre de la variable continua a graficar.  
bins (int): Número de bins para el histograma.  
kde (bool): Si True, dibuja la línea KDE.  
color (str): Color del histograma.

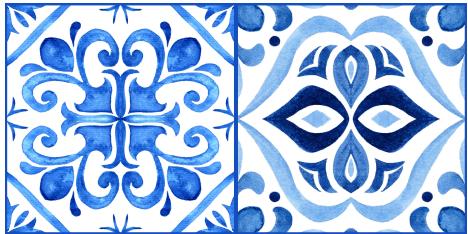
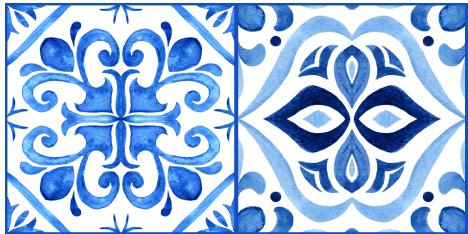
"""

```
plt.figure(figsize=(8, 5))  
sns.histplot(data=df, x=target, bins=bins, kde=kde,  
color=color)  
plt.title(f"Distribución de la variable continua: {target}")  
plt.xlabel(target)  
plt.ylabel("Frecuencia")  
plt.grid(axis="y", linestyle="--", alpha=0.6)  
plt.tight_layout()  
plt.show()
```

## DISTRIBUCION DEL PRECIO

- EL PRECIO PRESENTA UNA FUERTE ASIMETRÍA A LA DERECHA.
- SE APLICÓ LOG-TRANSFORMACIÓN PARA ESTABILIZAR LA VARIANZA Y MEJORAR EL AJUSTE DE LOS MODELOS.
- TAMBÍEN SE ELIMINARON LOS OUTLIERS EXTREMOS (>500 €, >800 €, >1000 €) EN DIFERENTES VERSIONES.





```
def get_features_num_regression(df,target_col,umbral_corr,pvalue=None,mostrar=False):
    """
    Conseguir la lista de features que tienen gran impacto en la target.
    Argumentos:
        df:DataFrame que se pasa de entrada
        target_col:la variable target con el que se quiere analizar,tiene que ser numerico
        umbral_corr: Umbral minimo para considerar una variable importante para el modelo
        pvalue: Certeza estadística con la que queremos validar la importancia de las feature
    Returns:
        Lista:Lista de features importantes.
        Mostrar: Muestra la matriz de correlación en una grafica HeatMap.
    """
    if target_col not in df.columns:
        raise ValueError(f"Columna target {target_col} no esta en el DataFrame dado.")
    if not isinstance(df, pd.DataFrame):
        raise TypeError("El dato de entrada tiene que ser un DataFrame.")

    if umbral_corr < 0 or umbral_corr > 1:
        raise ValueError("Umbral de correlacion tiene que estar entre 0 y 1")
    if pvalue is not None and (pvalue < 0 or pvalue > 1):
        raise ValueError("P-value tiene que estar entre 0 y 1")

    if not (np.issubdtype(df[target_col].dtype,np.number)):
        raise TypeError(f"Columna target {target_col} tiene que ser numerico amigo")

    cardinalidad_target = df[target_col].nunique() / len(df) * 100
    if cardinalidad_target < 10:
        warnings.warn(f"Columna target {target_col} tiene poca cardinalidad ({cardinalidad_target:.2f}%).")
    if cardinalidad_target > 95:
        warnings.warn(f"Columna target {target_col} tiene mucha cardinalidad ({cardinalidad_target:.2f}%).")
    if cardinalidad_target == 100:
        raise ValueError(f"Columna target {target_col} tiene 100% cardinalidad.")

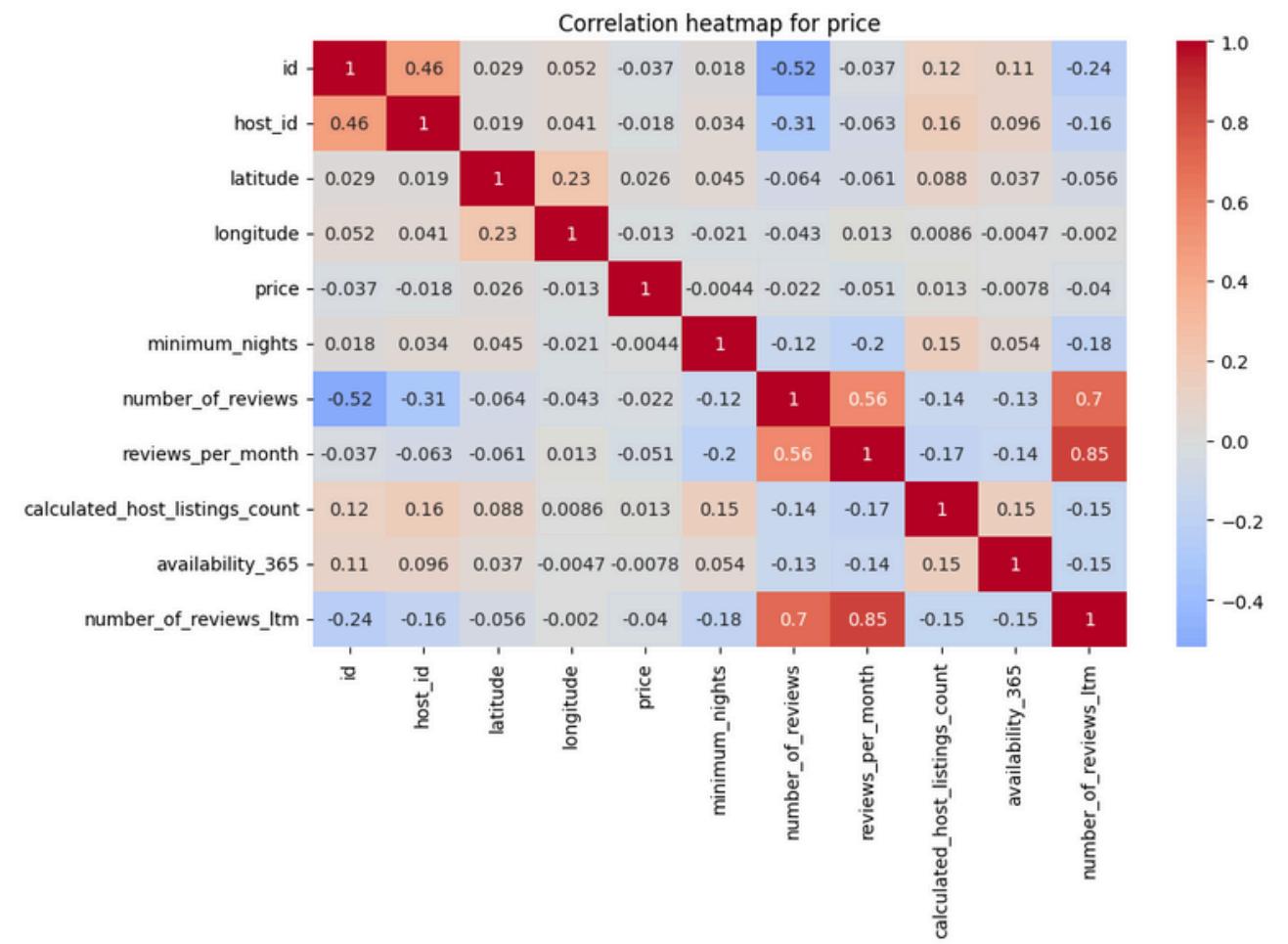
    if df[target_col].isnull().sum() > 0:
        raise ValueError(f"Columna target {target_col} tiene valores Nulos.")

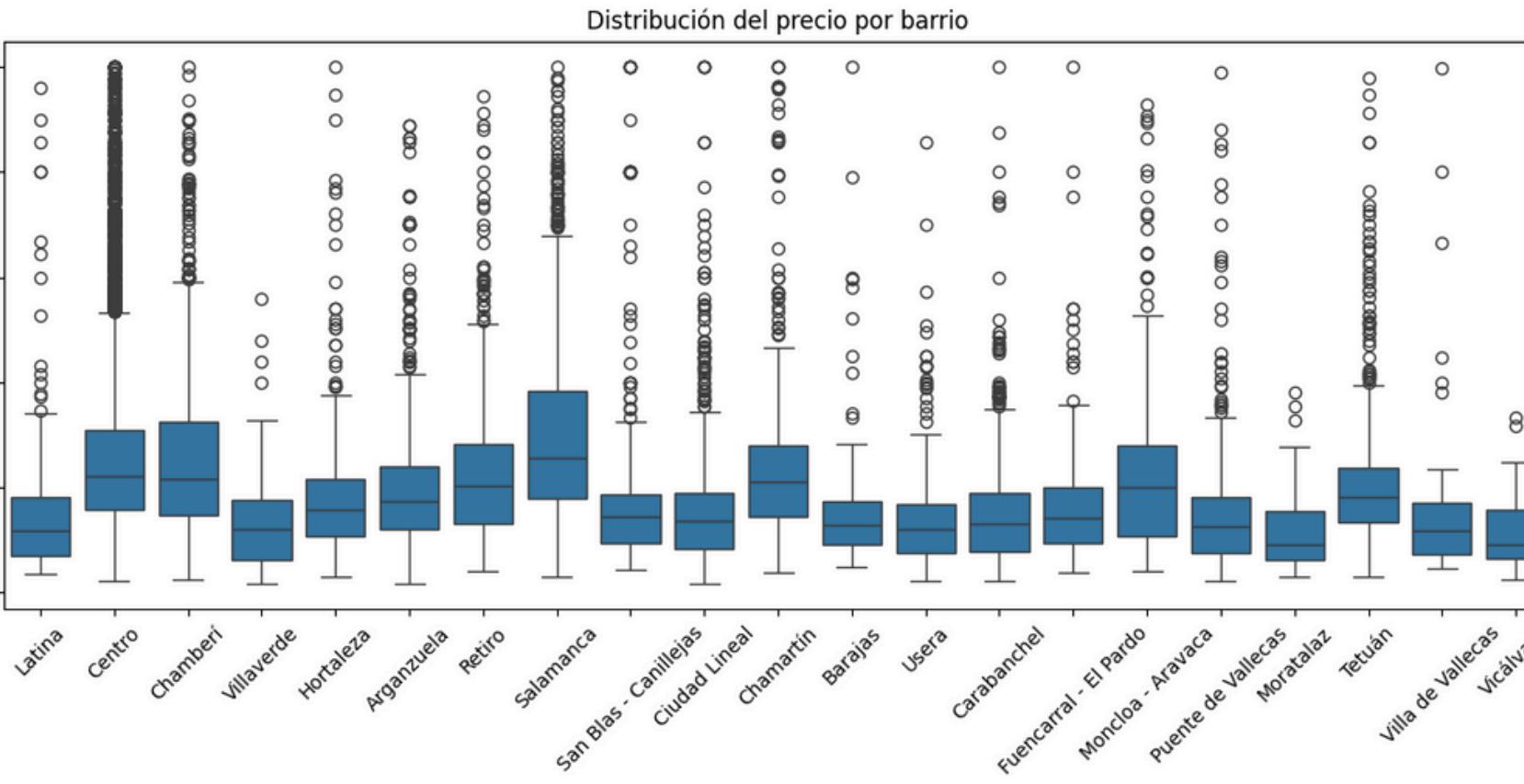
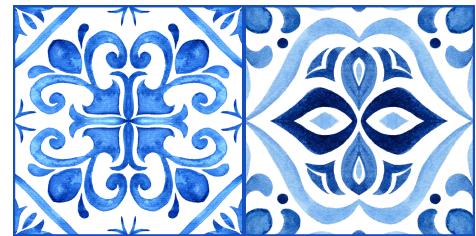
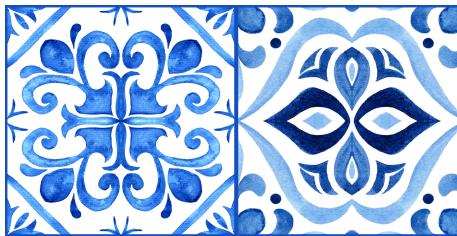
    corr = df.corr(numeric_only=True)[target_col]
    if mostrar:
        plt.figure(figsize=(10,6))
        sns.heatmap(df.corr(numeric_only=True),annot=True, cmap='coolwarm', center=0)
        plt.title(f"Correlation heatmap for {target_col}")
        plt.show()

    corr = corr[abs(corr) > umbral_corr]
    corr = corr.drop(target_col)
    lista = []
    if pvalue is not None:
        pvalues = []
        for col in corr.index:
            _, p = pearsonr(df[target_col], df[col])
            if p < pvalue:
                lista.append(col)
                pvalues.append(p)
        return lista, pvalues
    return lista
```

## VARIABLES MÁS PREDICTIVAS

- SE EVALUÓ LA CORRELACIÓN ENTRE VARIABLES NUMÉRICAS Y EL TARGET (PRICE).
- VARIABLES CON MAYOR RELACIÓN: MINIMUM\_NIGHTS, AVAILABILITY\_365, CALCULATED\_HOST\_LISTINGS\_COUNT.
- VARIABLES CATEGÓRICAS CLAVE: ROOM\_TYPE, NEIGHBOURHOOD.





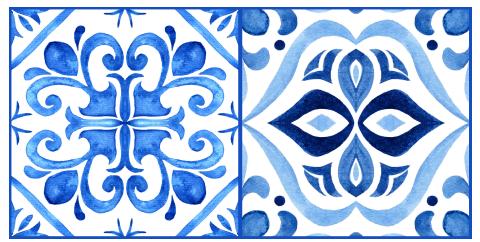
```
# BOXPLOT PRECIO POR BARRIO
PLT.FIGURE(FIGSIZE=(12, 6))
SNS.BOXPLOT(DATA=DF_PRICE_CLEAN, X="NEIGHBOURHOOD_GROUP", Y="PRICE")
PLT.XTICKS(ROTATION=45)
PLT.TITLE("DISTRIBUCIÓN DEL PRECIO POR BARRIO")
PLT.TIGHT_LAYOUT()
PLT.SHOW()
```

**AL ELIMINAR LOS PRECIOS EXTREMOS (FILTRANDO POR VALORES MENORES O IGUALES A 500 €), LA DISTRIBUCIÓN DEL PRECIO POR GRUPO DE BARRIO SE VISUALIZA DE FORMA MÁS CLARA.**

**SE OBSERVAN DIFERENCIAS NOTABLES ENTRE ZONAS: BARRIOS COMO CENTRO, SALAMANCA O RETIRO PRESENTAN MEDIANAS MÁS ALTAS QUE OTRAS ZONAS COMO USERA, VILLA DE VALLECAS O VICÁLVARO.**

**LA DISPERSIÓN DENTRO DE ALGUNOS BARRIOS ES SIGNIFICATIVA, LO QUE SUGIERE QUE HAY BASTANTE HETEROGENEIDAD INCLUSO DENTRO DE UNA MISMA ZONA.**

**ESTA VARIABLE (NEIGHBOURHOOD\_GROUP) APORTA INFORMACIÓN RELEVANTE Y SERÁ ÚTIL COMO PREDICTORA EN EL MODELO.**



```
def get_features_num_regression(df,target_col,umbral_corr,pvalue=None,mostrar=False):
    """
    Conseguir la lista de features que tienen gran impacto en la target.
    Argumentos:
        df:DataFrame que se pasa de entrada
        target_col:la variable target con el que se quiere analizar,tiene que ser numerico
        umbral_corr: Umbral minimo para considerar una variable importante para el modelo
        pvalue: Certeza estadística con la que queremos validar la importancia de las feature
    Returns:
        Lista:Lista de features importantes.
        Mostrar: Muestra la matriz de correlación en una grafica HeatMap.
    """
    if target_col not in df.columns:
        raise ValueError(f"Columna target {target_col} no esta en el DataFrame dado.")
    if not isinstance(df, pd.DataFrame):
        raise TypeError("El dato de entrada tiene que ser un DataFrame.")

    if umbral_corr < 0 or umbral_corr > 1:
        raise ValueError("Umbral de correlacion tiene que estar entre 0 y 1")
    if pvalue is not None and (pvalue < 0 or pvalue > 1):
        raise ValueError("P-value tiene que estar entre 0 y 1")

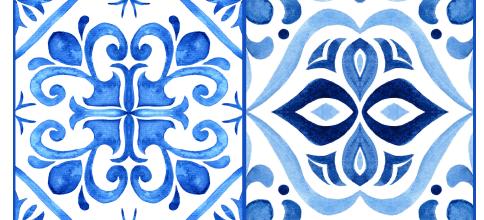
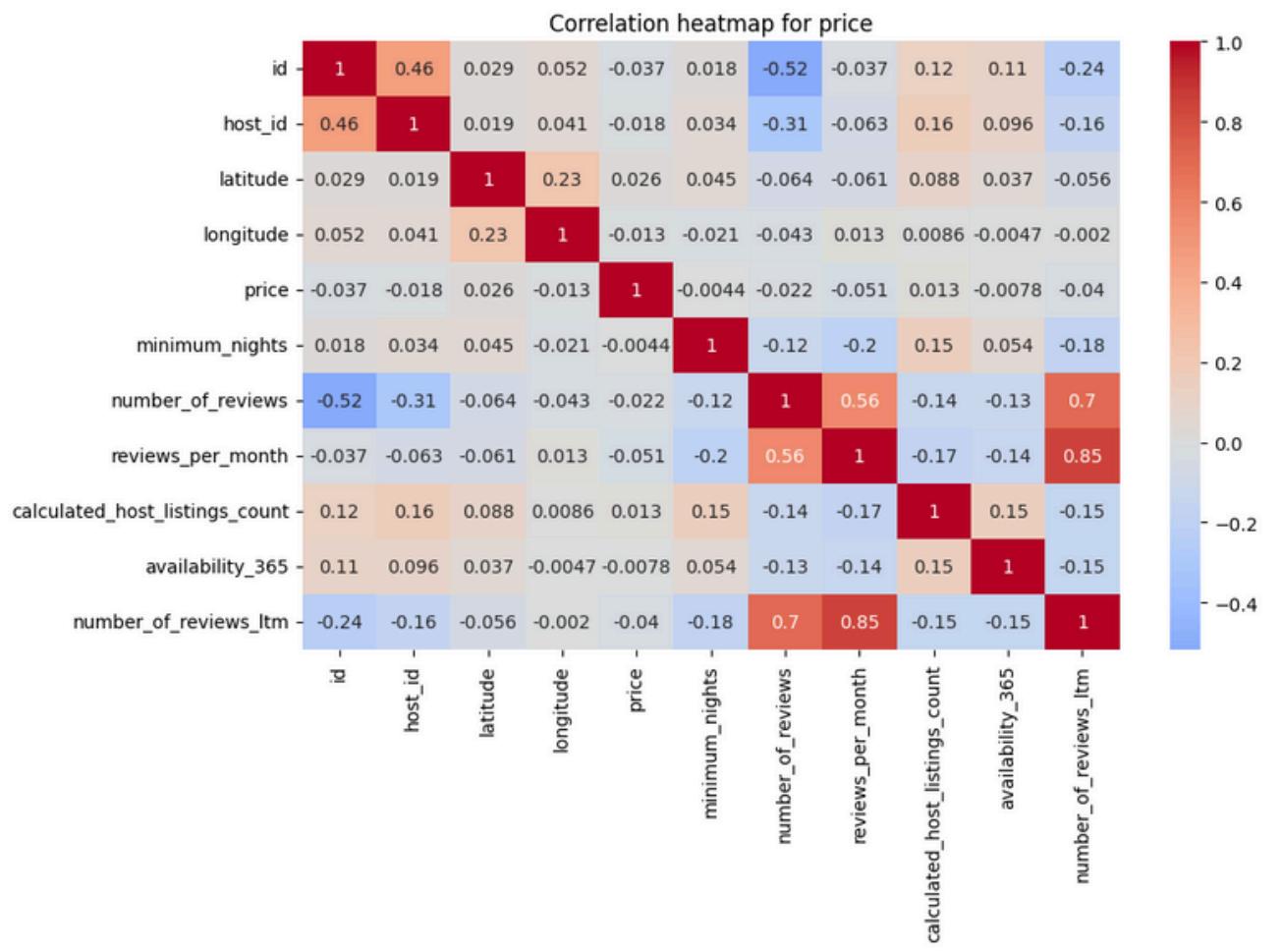
    if not (np.issubdtype(df[target_col].dtype,np.number)):
        raise TypeError(f"Columna target {target_col} tiene que ser numerico amigo")

    cardinalidad_target = df[target_col].nunique() / len(df) * 100
    if cardinalidad_target < 10:
        warnings.warn(f"Columna target {target_col} tiene poca cardinalidad ({cardinalidad_target:.2f}%).")
    if cardinalidad_target > 95:
        warnings.warn(f"Columna target {target_col} tiene mucha cardinalidad ({cardinalidad_target:.2f}%).")
    if cardinalidad_target == 100:
        raise ValueError(f"Columna target {target_col} tiene 100% cardinalidad.")

    if df[target_col].isnull().sum() > 0:
        raise ValueError(f"Columna target {target_col} tiene valores Nulos.")

    corr = df.corr(numeric_only=True)[target_col]
    if mostrar:
        plt.figure(figsize=(10,6))
        sns.heatmap(df.corr(numeric_only=True),annot=True, cmap='coolwarm', center=0)
        plt.title(f"Correlation heatmap for {target_col}")
        plt.show()

    corr = corr[abs(corr) > umbral_corr]
    corr = corr.drop(target_col)
    lista = []
    if pvalue is not None:
        pvalues = []
        for col in corr.index:
            _, p = pearsonr(df[target_col], df[col])
            if p < pvalue:
                lista.append(col)
                pvalues.append(p)
    return lista
```



# PROCESO DE MODELADO

CODIFICACIÓN DE  
VARIABLES  
CATEGÓRICAS  
(GET\_DUMMIES)

ESCALADO DE  
VARIABLES  
NUMÉRICAS  
(STANDARDSCALER)

TRAIN/TEST SPLIT  
(80/20)

- RANDOM FOREST  
(BASELINE Y  
AJUSTADO)  
XGBOOST

APLICACIÓN DE  
 $\log(\text{PRICE})$  Y FILTROS  
POR OUTLIERS (<1000  
€, <800 €, <500 €)



# COMPARATIVA DE RESULTADOS



MEJORA SIGNIFICATIVA AL CENTRARSE EN PRECIOS REALISTAS Y APLICAR LOGARITMO.

EL MODELO FINAL PREDICE CON PRECISIÓN RAZONABLE Y BAJO ERROR PARA ALOJAMIENTOS TÍPICOS.

## SIN FILTRAR

MAE: 65.84€

RMSE: 209.09€

R<sup>2</sup>: -0.02



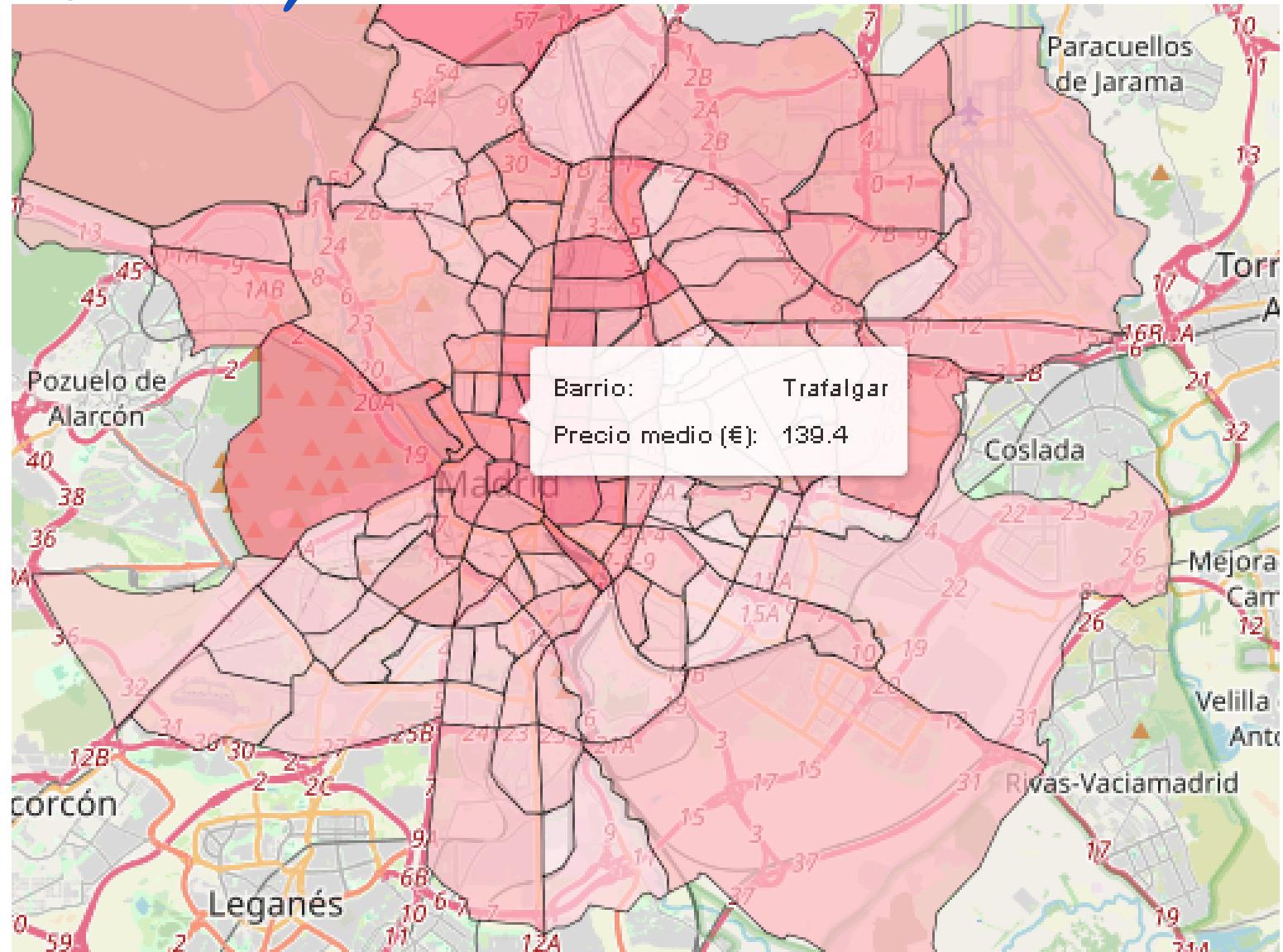
## LOG(PRICE) + FILTRO < 500 €

MAE: 32.94 €

RMSE: 55.93 €

R<sup>2</sup>: 0.40

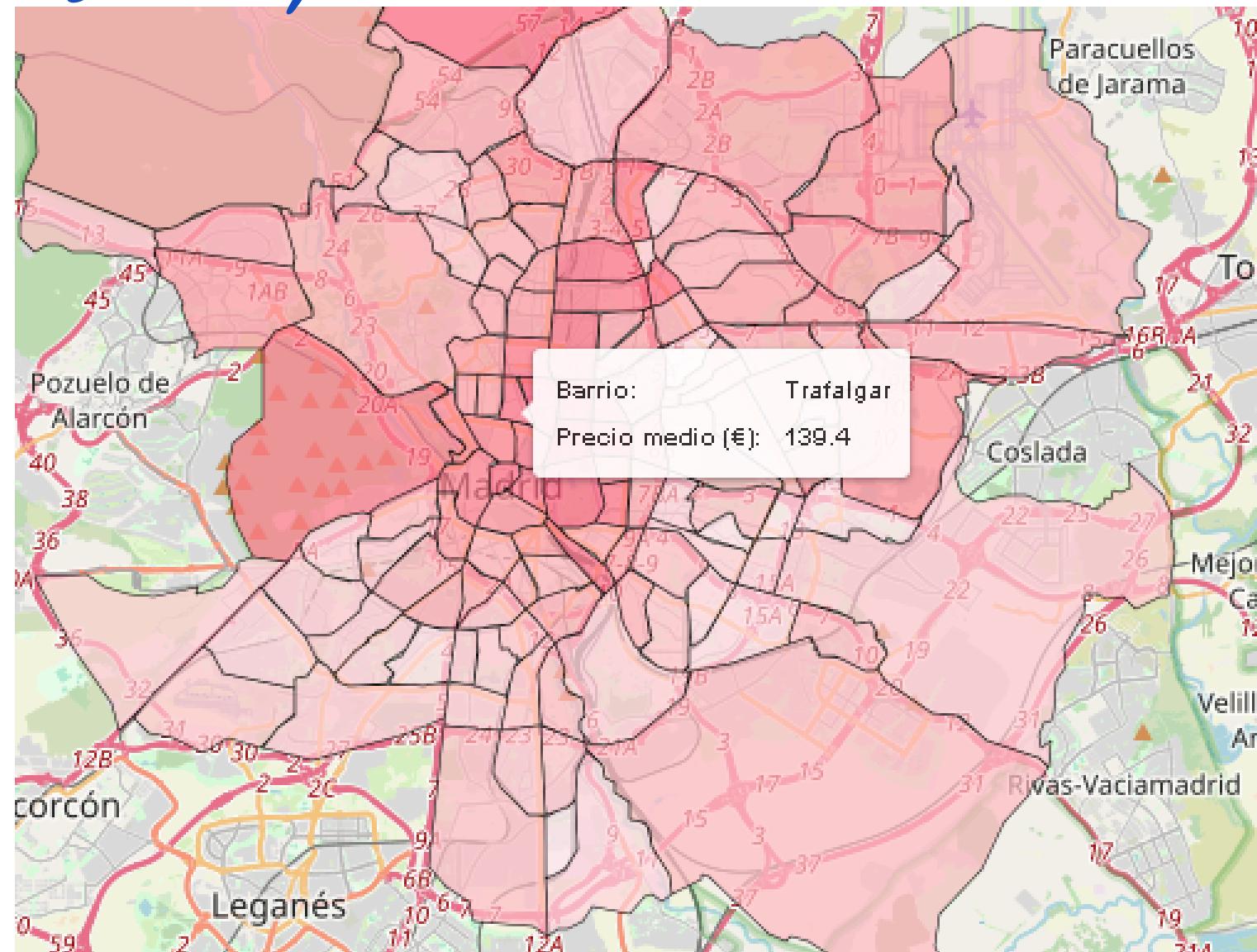
# Mapa de Folium



## VISUALIZACIÓN: PRECIOS MEDIOS POR BARRIOS

- MAPA DE FOLIUM CON MEDIA DE PRECIOS POR BARRIO
- TONOS DESDE ROSA CLARO A CORAL (INSPIRADO EN BRANDING AIRBNB)
- INSIGHT: BARRIOS CÉNTRICOS MÁS CAROS, ZONAS PERIFÉRICAS MÁS BARATAS

# Mapa de Folium



```
# Cargar archivo GeoJSON
with open("src/data/madrid.geojson", "r", encoding="utf-8") as f:
    geojson_data = json.load(f)

# Calcular precios medios por barrio (filtrando a precios <= 500)
barrio_precio = df_price_clean.groupby("neighbourhood")["price"].mean().to_dict()

# Añadir el precio medio al GeoJSON
for feature in geojson_data["features"]:
    nombre = feature["properties"]["name"]
    precio = barrio_precio.get(nombre, None)
    feature["properties"]["precio_medio"] = round(precio, 2) if precio else None

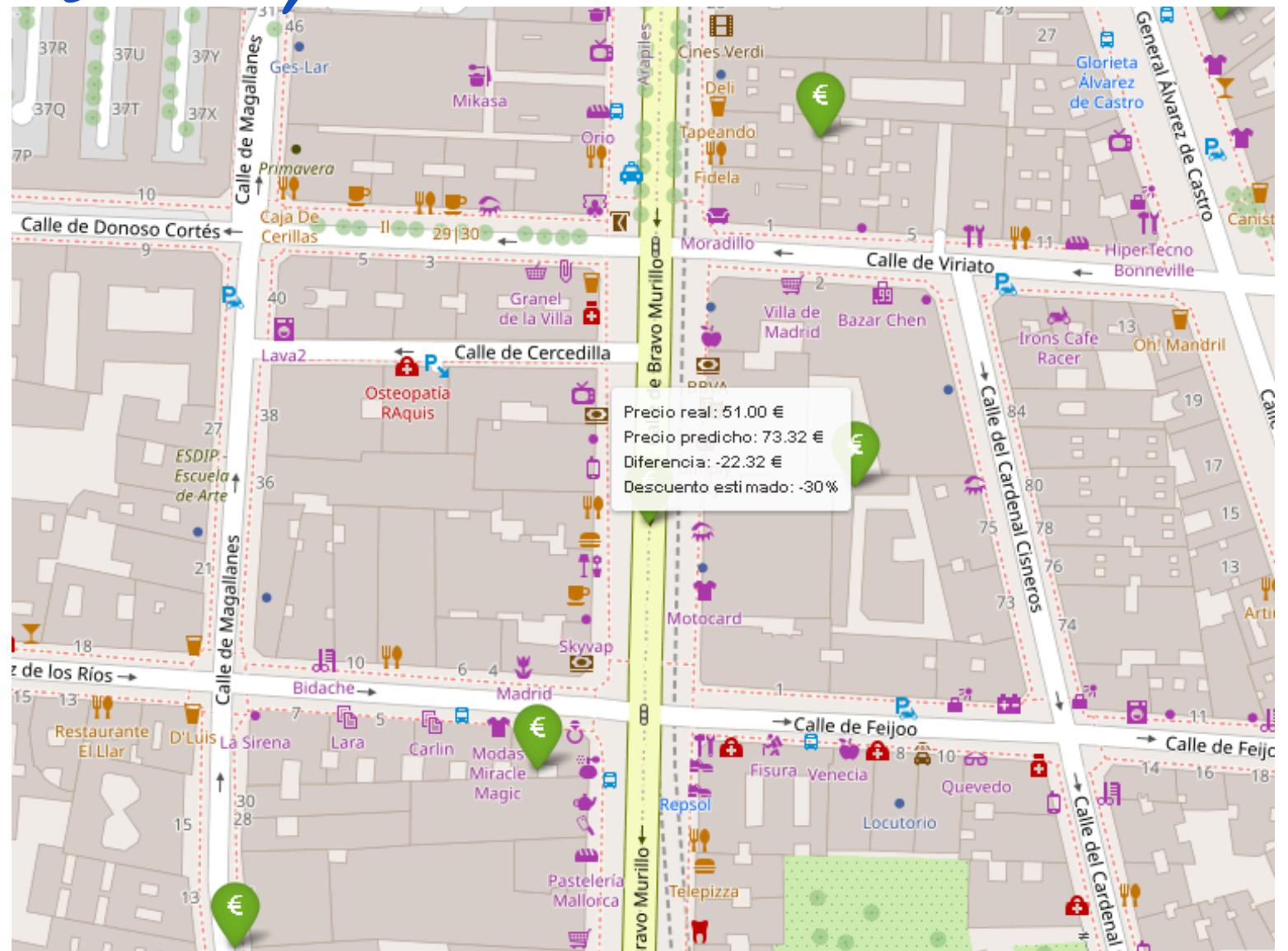
# Crear escala de color basada en los valores reales
precios = [v for v in barrio_precio.values() if v is not None]
colormap = cm.LinearColormap(
    colors=["#FFE5EC", "#FF8C94", "#FF1F5B"], # de rosa muy claro a coral intenso
    vmin=min(precios),
    vmax=max(precios),
    caption="Precio medio por barrio (€)"
)

# Crear el mapa base
m = folium.Map(location=[40.4168, -3.7038], zoom_start=11)

# Añadir la capa GeoJSON
folium.GeoJson(
    geojson_data,
    name="Precio medio por barrio",
    style_function=lambda feature:
        {
            "fillColor": colormap(feature["properties"]["precio_medio"]) if feature["properties"]["precio_medio"] else "lightgray",
            "color": "black",
            "weight": 0.5,
            "fillOpacity": 0.7,
        },
    tooltip=folium.GeoJsonTooltip(
        fields=["name", "precio_medio"],
        aliases=["Barrio:", "Precio medio (€):"],
        localize=True
    )
).add_to(m)

# Añadir leyenda al mapa
colormap.add_to(m)
```

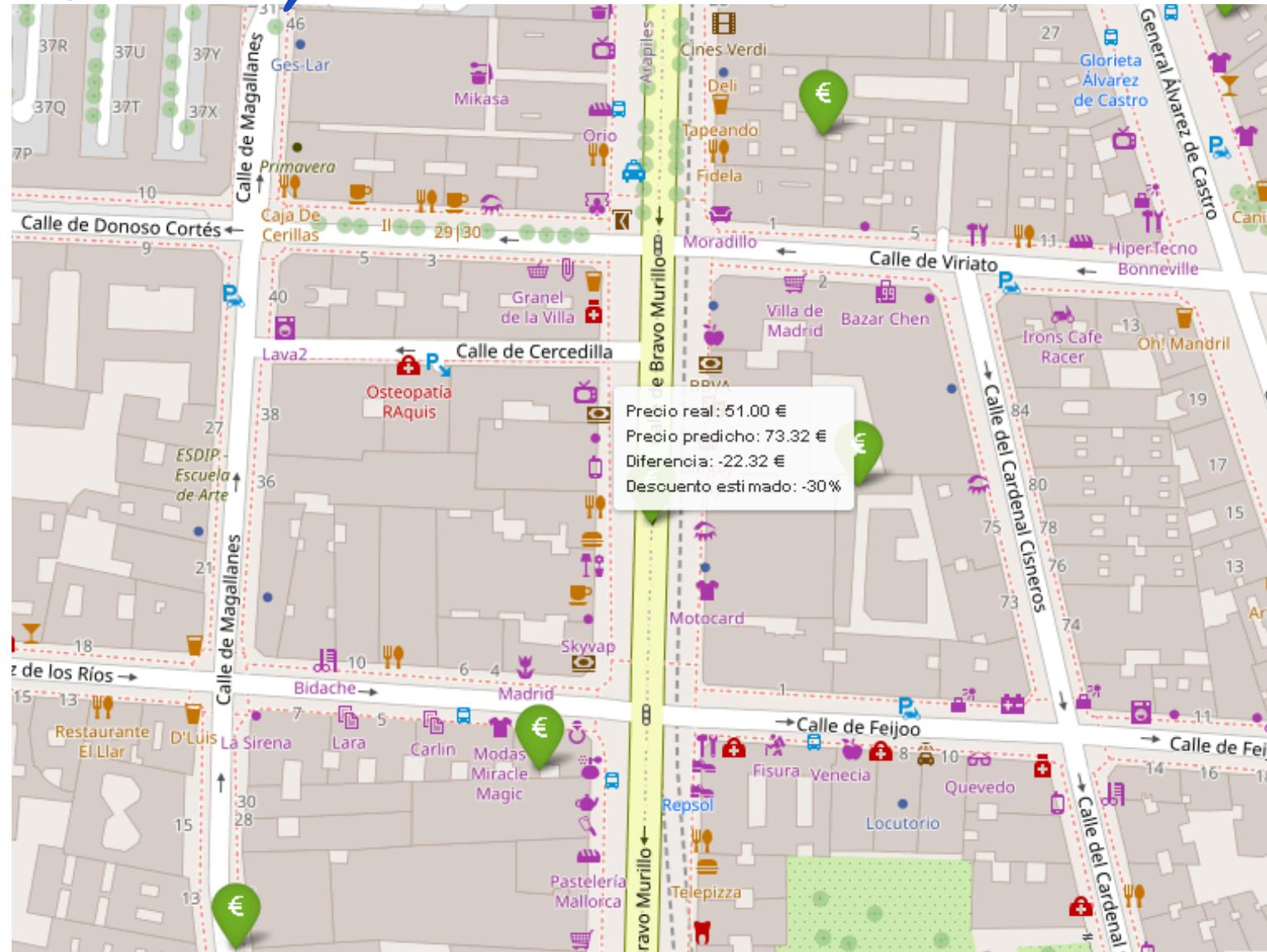
# Mapa de Folium



## VISUALIZACIÓN: OPORTUNIDADES DE INVERSIÓN

- DETECCIÓN DE ALOJAMIENTOS CON PRECIO REAL  
>20 % POR DEBAJO DEL ESTIMADO
- REPRESENTACIÓN EN MAPA CON MARCADORES  
VERDES
- EXPORTADO COMO CSV  
(OPORTUNIDADES\_AIRBNB\_BARATOS.CSV)

# Mapa de Folium



```
# Añadir coordenadas al dataframe de resultados
df_resultados_con_coords = df_resultados.copy()
df_resultados_con_coords[["latitude", "longitude"]] = df_madrid.loc[df_resultados.index, ["latitude", "longitude"]]
```

```
# Filtrar oportunidades con coordenadas
oportunidades_geo = df_resultados_con_coords[
    (df_resultados_con_coords["es_oportunidad"]) &
    (df_resultados_con_coords["latitude"].notnull()) &
    (df_resultados_con_coords["longitude"].notnull())
]
```

```
# Crear mapa base
m_oportunidades = folium.Map(location=[40.4168, -3.7038], zoom_start=12)
cluster = MarkerCluster().add_to(m_oportunidades)
```

```
# Añadir marcadores
for _, row in oportunidades_geo.iterrows():
    tooltip_text = (
        f"Precio real: {row['price_real']:.2f} €  
"
        f"Precio predicho: {row['price_predicho']:.2f} €  
"
        f"Diferencia: {row['diferencia']:.2f} €  
"
        f"Descuento estimado: {row['%_por_deabajo']:.0%}"
    )
    folium.Marker(
        location=[row["latitude"], row["longitude"]],
        tooltip=tooltip_text,
        icon=folium.Icon(color="green", icon="euro-sign", prefix="fa")
    ).add_to(cluster)
```

m\_oportunidades

# CONCLUSIONES Y FUTURAS MEJORAS

- EL MODELO PREDICE CON BUENA PRECISIÓN PARA EL RANGO NORMAL DE PRECIOS
- SE HA DEMOSTRADO APLICABILIDAD PRÁCTICA (DETECCIÓN DE CHOLLOS)

## MEJORAS POSIBLES:

- INCLUIR MÁS VARIABLES DEL ANUNCIO (FOTOS, AMENITIES, VALORACIONES)
- MEJORAR REPRESENTACIÓN GEOESPACIAL (LAT/LONG, DISTANCIA A PUNTOS CLAVE)
- DESPLEGAR COMO APP PARA USUARIOS
- ENTRENAR UN MODELO ESPECÍFICO POR CADA TIPO DE ALOJAMIENTO `ROOM\_TYPE`



**¡GRACIAS POR TU ATENCIÓN!**

*Sandra López - Julio 2025*