Proyecto Mimir -

Equipo 19 "Se cayo el Internet"

'Mímir, un gigante sabio de la mitología nórdica. Mímir era conocido por su inmenso conocimiento y por beber del pozo de la sabiduría, ubicado en las raíces del árbol del mundo, Yggdrasil. Se dice que quien bebía del pozo adquiría un conocimiento profundo y la capacidad de ver el pasado, el presente y el futuro. Mímir fue consultado en varias ocasiones por los dioses nórdicos para obtener consejos y conocimiento sobre eventos futuros.'

Documentación:

En la actualidad, el campo de la inteligencia artificial ha alcanzado niveles sorprendentes en áreas como el procesamiento de texto y de imágenes. Sin embargo, nuestro equipo ha enfocado sus esfuerzos en un aspecto particularmente relevante: la implementación de inteligencia artificial en el ámbito de los datos geográficos.

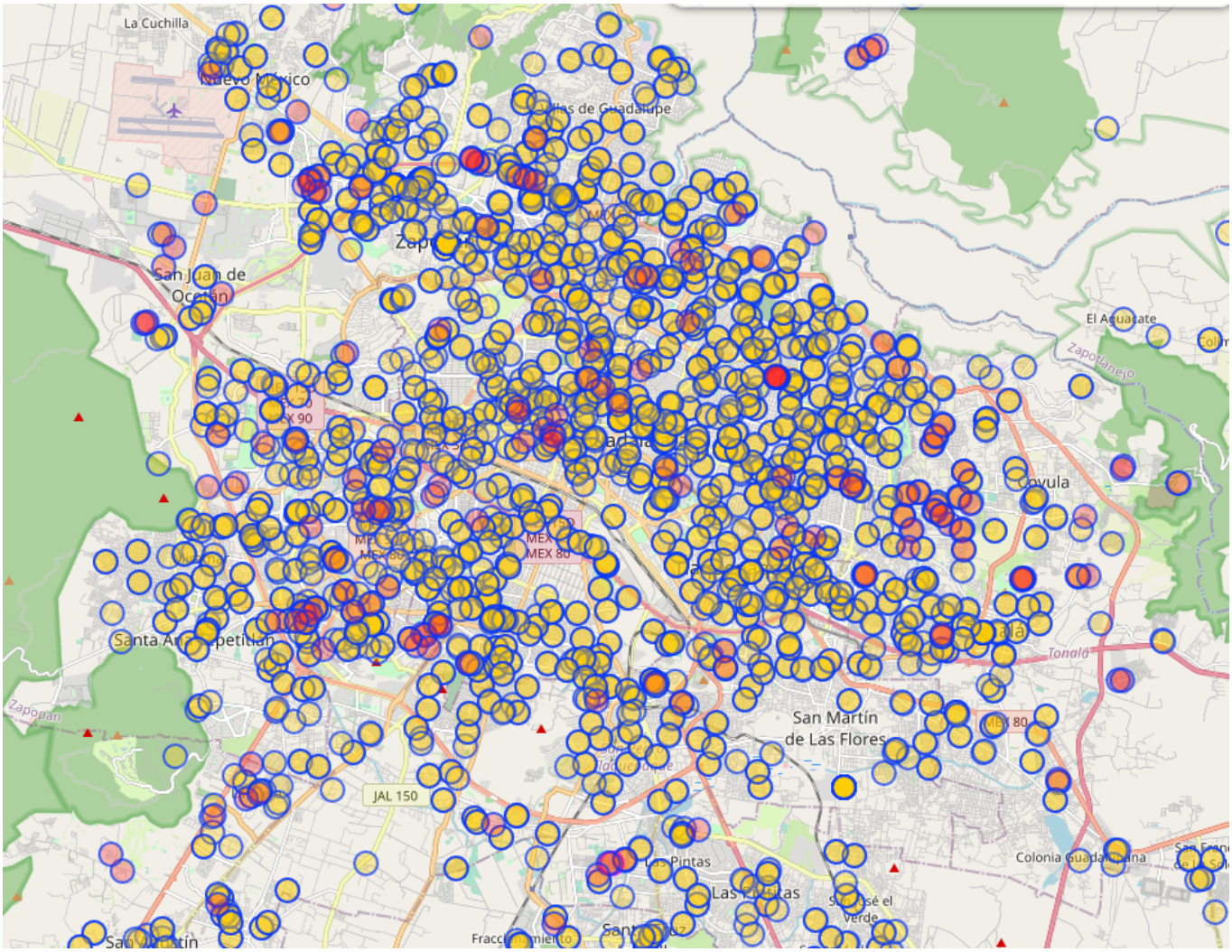
Nuestro proyecto se basa en un modelo de aprendizaje automático que permite realizar predicciones precisas sobre los precios de los inmuebles a partir de información geográfica. Este enfoque tiene como objetivo fundamental mejorar el análisis de los mercados inmobiliarios, impulsar y optimizar los sistemas de tasación de créditos hipotecarios en los bancos, así como perfeccionar los esquemas estadísticos tanto en el sector privado como en el público.

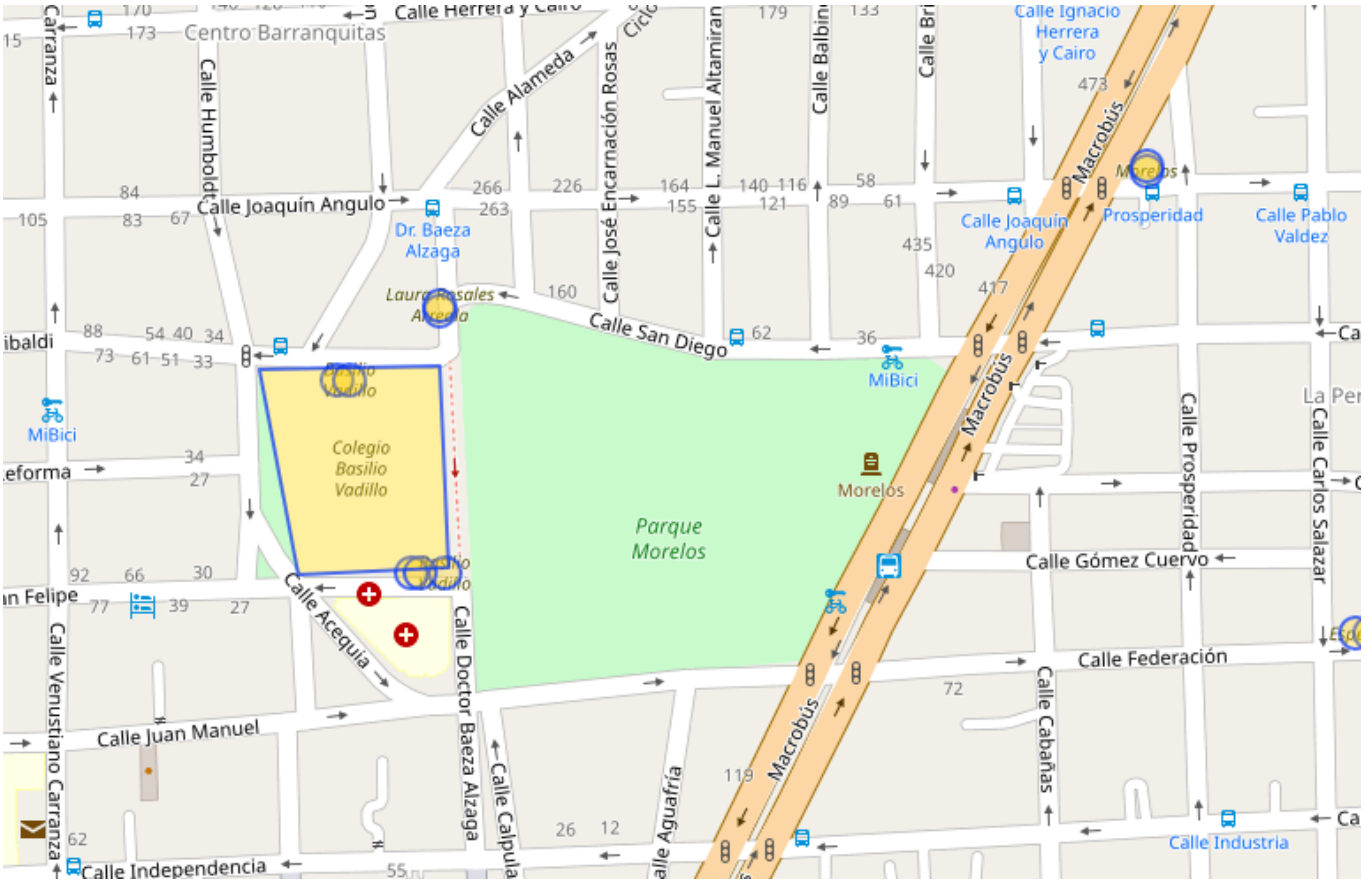
Al combinar la potencia del aprendizaje automático con los datos geográficos, estamos abriendo nuevas puertas y generando insights valiosos que antes parecían inalcanzables. Nuestra tecnología no solo proporciona una visión más precisa y detallada de los precios de los inmuebles, sino que también aporta una perspectiva vanguardista para el desarrollo estratégico de los sectores inmobiliario y financiero.

Rascado de datos:

Se realizo un rascado masivo de la API de OpenStreetMaps, (Overpassturbo), con el objetivo de extraer información geográfica valiosa, esto con el objetivo de tener un mapeo significativo de los inmuebles de la zona metropolitana de Guadalajara.

En este caso los archivos se extrajeron en formato Geojson.


La Cuchilla
Nuevo México
Villas de Guadalupe
San Juan de Ocotán
El Aguacate
Zapotlanejo
Copula
Tonalá
Santa Ana Tepetitlán
Zapopan
San Martín de Las Flores
JAL 150
Las Pintas
Las Huertas
Colonia Guadalajara
Fraccionamiento



Ademas se realizo una extracción de datos por medio de un webscraper de diversas paginas de inmobiliarias con el objetivo de obtener un dataset de precios de inmuebles, en el cual se pudiera hacer una entrenamiento con dichos datos. (Estos datos se guardaron en csv y se procesaron por un tratamiento de datos por medio de pandas).

Variables del dataset:

1. web-scraper-order
2. web-scraper-url
3. price
4. bedrooms
5. bathrooms
6. area
7. address
8. zip-code
9. colony
10. city
11. state
12. country
13. latitud
14. longitud

Aumentación de los datos del dataset

(En proceso aún).

Modelo:

Utilizamos Python como lenguaje de programación y se hace uso de varias bibliotecas y herramientas de Machine Learning para realizar análisis de datos geográficos y predecir los precios de los inmuebles. A continuación, se muestra una documentación de las bibliotecas principales utilizadas:

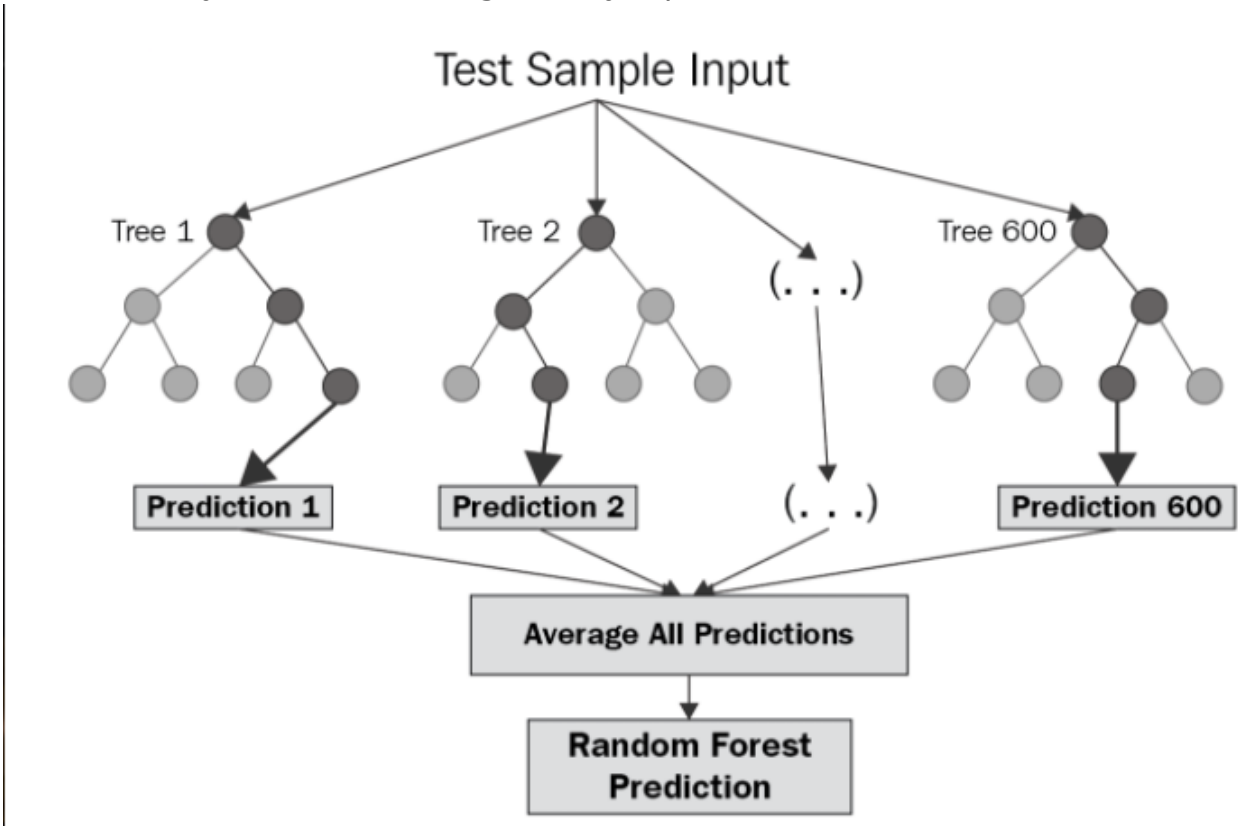
- DecisionTreeClassifier de sklearn.tree: Modelo de árbol de decisión utilizado para clasificación.
- RandomForestRegressor de sklearn.ensemble: Modelo de regresión de bosque aleatorio utilizado para predecir los precios de los inmuebles.
- xgb de xgboost: Implementación de XGBoost, un algoritmo de aprendizaje automático basado en árboles que proporciona un mejor rendimiento y escalabilidad.

RandomForest

Es una combinación de árboles predictores (los arboles de decisión) tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos. Esto quiere decir que para cada arbol del forest se construye de la siguiente forma:

1. Sea *N* el número de casos de prueba, *M* es el número de variables en el clasificador.
2. Sea *m* el número de variables de entrada a ser usado para determinar la decisión en un nodo dado; *m* debe ser mucho menor que *M*
3. Elegir un conjunto de entrenamiento para este árbol y usar el resto de los casos de prueba para estimar el error.
4. Para cada nodo del árbol, elegir aleatoriamente *m* variables en las cuales basar la decisión. Calcular la mejor partición del conjunto de entrenamiento a partir de las *m*

variables. (Adjuntamos una imagen de ejemplo)



Nota: Instalar todo sobre las siguientes dependencias :DDDD

```
pip install pandas numpy seaborn matplotlib scikit-learn xgboost
```

Pasos:

1. Primero se realizo la lectura de los datos del csv ya procesado y tradado.
2. Filtro de datos
   Luego se ordenaron los precios de los inmuebles y se eliminanaron los valores atípicos. Solo se mantienen los inmuebles cuyos precios se encuentran dentro del 5% y el 95% de los valores ordenados.

   (La justificación de esta decisión fue tomada ya que en la gráfica de distribución de los precios se tenían problemas con las colas, sobre todo con los inmuebles de mayor valor, esto ya que empeoraban el modelo al considerar casas muy caras ya sea por otros features que no teníamos a la mano, pero claramente si se hace el analisis exploratorio, se puede observar esos detalles. (cosas que un estadistico vería xD)).
3. StandardScaler para estandarizar las características numéricas del dataset. Luego, se escala y se seleccionan las características explicativas (número de habitaciones, baños, área, latitud y longitud) y la variable objetivo (precio).

Nota: Se esta considerando la aumentación de los datos, pero a estas horas todavía no esta completamente implementado. Espero mas tarde que lo este.

4. Se crea un histograma para visualizar la distribución de los precios de los inmuebles.
5. División del conjunto de datos:

```
X_train, X_test, y_train, y_test = train_test_split(x_explicativa,
Y_respuesta, test_size=0.2, random_state=42)
```

Se divide el conjunto de datos en conjuntos de entrenamiento y prueba, utilizando el 80% de los datos para entrenar el modelo y el 20% para evaluar su rendimiento. (El poderoso train-test)

Nota: Por temas de tiempo reconozco que no se debería de utilizar el SimpleImputer para imputar los valores faltantes en los conjuntos de entrenamiento y prueba, pero esta es una solución rapida de momento. Mas adelante se podría considerar otra forma.

Creación y entrenamiento del modelo

Primero se hizo la creación de un modelo de Random Forest utilizando la herramienta GridSearchCV para la búsqueda de hiperparámetros óptimos.

Nota: (El Mentor Said me lo comento y esto nos ayudo mucho :DDDD muchas gracias).

Esto se utilizo para realizar una búsqueda exhaustiva de los mejores hiperparámetros para el modelo de Random Forest. Los hiperparámetros que se están ajustando son:

- n_estimators: el número de árboles en el bosque.
- max_depth: la profundidad máxima de cada árbol en el bosque.
- min_samples_split: el número mínimo de muestras requeridas para
- dividir un nodo interno.
- min_samples_leaf: el número mínimo de muestras requeridas para ser una hoja.

Se proporcionan diferentes opciones para cada hiperparámetro y GridSearchCV probará todas las combinaciones posibles para determinar cuál da el mejor rendimiento utilizando la validación cruzada.

Esto se hizo previo al entrenamiento del modelo.

Una vez que el GridSearch cumple con su chamba, se determino que el mejor modelo para este caso de 253 casas, era un randomforest de 1000 arboles, de profundidad maxima 6, a un ratio de aprendizaje de 0.1.

```
modelo = xgb.XGBRegressor(n_estimators=1000, max_depth=6, learning_rate=0.1,
subsample=0.8, colsample_bytree=0.8, random_state=42)
```

Con todo esto se procedio a entrenar el modelo. Una vez que el modelo se entreno se le pidio que realizara las predicciónes sobre el conjunto de prueba X_test, con lo cual se hizo Cross-validation.

Mean absolute porcentual error

El MAPE mide el promedio del porcentaje absoluto de error entre las predicciones y los valores reales. El MAPE es una métrica útil para evaluar el rendimiento del modelo de Random Forest en términos de la precisión de las predicciones. Un MAPE más bajo indica que el modelo tiene un menor error porcentual medio, lo que implica una mayor precisión en las predicciones.

Resultados:

Una vez que se entreno el modelo y se hicieron las predicciones sobre el conjunto de prueba, y ademas se calculo el error absoluto medio procentual, se obtuvo un MAPE aproximadamente de $0.332443793...$, esto a pesar de que simplemente se trabajo con un dataset de 253 datos y solo se consideraron (de momento) 5 features (variables explicativas).

Esto es muy bueno, pese a que solo se estuvieron trabajando con un numero chiquito de datos. Con mas tiempo y desarrollo estamos convencidos como equipo de que el modelo tendría mejor rendimiento con muestras mas grandes y mayores parametros como variables explicativas. SIendo así un modelo generativo de precios de inmuebles. (Como si tuvieramos un oraculo de precios :D)

(De momento, tenemos este resultado, estamos esperando la prueba de la aumentacion de los datos)

Sobre el despliegue...