INFORME DE EVALUACIÓN Y PROPUESTAS

Proyecto Semestral de Geoinformática

Análisis Personalizado y Recomendaciones
Universidad de Santiago de Chile

Estudiante: Byron Caices
Profesor: Dr. Francisco Parra O.
Área de Interés: Negocios y Urbanismo
Enfoque: Análisis Inmobiliario y Plusvalía

Agosto 2025

Índice

1. Resumen Ejecutivo

ANÁLISIS DEL PERFIL

Byron presenta un **perfil ideal** para proyectos de geoinformática aplicada al sector inmobiliario:

- **Fortaleza técnica:** Dominio de Python y SQL, fundamentales para análisis geoespacial
- **Sin experiencia previa en GIS:** Oportunidad de aprendizaje significativo
- **Interés comercial claro:** Enfoque en negocios y urbanismo con aplicación práctica
- **Visión específica:** Análisis de plusvalía y factores urbanos

Recomendación: Desarrollar un proyecto que combine machine learning espacial con análisis del mercado inmobiliario chileno, aprovechando su experiencia en Python para crear una solución innovadora y comercialmente viable.

2. Análisis del Perfil del Estudiante

2.1. Competencias Técnicas

Habilidad	Nivel	Aplicación en Geoinformática
Python	Avanzado	GeoPandas, Rasterio, Scikit-learn para ML espacial
SQL	Intermedio	PostGIS para consultas espaciales complejas
JavaScript	Básico	Google Earth Engine, Leaflet para visualización
R	Básico	Análisis estadístico espacial complementario
Datos Geográficos	Principiante	Curva de aprendizaje acelerada esperada

2.2. Áreas de Interés y Potencial

2.2.1. Negocios y Geomarketing

- Análisis de localización óptima para retail
- Predicción de éxito comercial basado en variables espaciales
- Segmentación de mercado geográfica
- Optimización de rutas de distribución

2.2.2. Urbanismo y Plusvalía

- Modelado de precios inmobiliarios
- Análisis de factores de valorización
- Predicción de desarrollo urbano

- Evaluación de impacto de infraestructura

3. Propuesta de Proyecto 1: Sistema Inteligente de Valoración Inmobiliaria

3.1. Descripción General

Desarrollo de una plataforma que utiliza machine learning espacial para predecir y analizar el valor de propiedades en el Gran Santiago, considerando múltiples variables urbanas y su evolución temporal.

3.2. Arquitectura Técnica

```python
import geopandas as gpd
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
import folium
from shapely.geometry import Point
import osmnx as ox

class RealEstateValuationSystem:
    def __init__(self, study_area):
        self.study_area = study_area
        self.properties = gpd.GeoDataFrame()
        self.urban_features = {}
        self.model = None

    def collect_property_data(self):
        """Recolecta datos de portales inmobiliarios"""
        # Web scraping de portalinmobiliario.com
        # Geocodificaci n de direcciones
        # Estructuraci n en GeoDataFrame
        pass

    def extract_urban_features(self):
        """Extrae caracter sticas urbanas relevantes"""
        features = {}

        # 1. Accesibilidad a metro
        metro_stations = gpd.read_file('metro_santiago.geojson')
        features['dist_metro'] = self.calculate_nearest_distance(
            self.properties, metro_stations
        )

        # 2.  reas  verdes (NDVI desde Sentinel-2)
        features['green_index'] = self.calculate_ndvi_buffer(
            self.properties, radius=500
        )

        # 3. Densidad comercial (POIs desde OSM)
```

```
        features['commercial_density'] = self.
            calculate_poi_density(
            categories=['shop', 'restaurant', 'bank']
        )

        # 4. Calidad del aire (datos SINCA)
        features['air_quality'] = self.interpolate_air_quality
            ()

        # 5.  ndice  de seguridad (datos de delitos)
        features['safety_index'] = self.calculate_safety_score
            ()

        # 6. Conectividad vial
        G = ox.graph_from_place('Santiago, Chile', network_type
            ='drive')
        features['street_connectivity'] = self.
            calculate_connectivity(G)

        return features

    def train_valuation_model(self):
        """Entrena modelo de valoraci n con validaci n
            espacial"""
        X = pd.DataFrame(self.urban_features)
        y = self.properties['price_per_m2']

        # Agregar coordenadas para autocorrelaci n espacial
        X['lat'] = self.properties.geometry.y
        X['lon'] = self.properties.geometry.x

        # Split con consideraci n espacial
        X_train, X_test, y_train, y_test = self.
            spatial_train_test_split(
            X, y, test_size=0.2
        )

        # Modelo con regularizaci n espacial
        self.model = GradientBoostingRegressor(
            n_estimators=500,
            max_depth=8,
            learning_rate=0.01,
            subsample=0.8,
            min_samples_split=10,
            random_state=42
        )

        self.model.fit(X_train, y_train)

        # Feature importance
        self.analyze_feature_importance()

        return self.model

    def predict_property_value(self, address):
        """Predice valor de propiedad nueva"""
        location = self.geocode_address(address)
        features = self.extract_features_for_point(location)
```

```
90
91          prediction = self.model.predict([features])[0]
92          confidence_interval = self.
                calculate_confidence_interval(features)
93
94          return {
95              'predicted_value': prediction,
96              'confidence_interval': confidence_interval,
97              'main_factors': self.explain_prediction(features)
98          }
99
100     def generate_heatmap(self):
101         """Genera mapa de calor de valores"""
102         # Crear grid hexagonal
103         hexagons = self.create_hexgrid(self.study_area)
104
105         # Predecir valor para cada hex gono
106         for hex in hexagons:
107             centroid = hex.centroid
108             features = self.extract_features_for_point(centroid
                    )
109             hex['predicted_value'] = self.model.predict([
                    features])[0]
110
111         # Visualizar con Folium
112         return self.create_interactive_map(hexagons)
```

Listing 1: Pipeline principal del sistema

3.3. Fuentes de Datos

Dato	**Fuente**	**Procesamiento**	**Frecuencia**
Precios inmobiliarios	PortalInmobiliario.com	Web scraping + geocoding	Diaria
Transporte público	DTPM / Metro Santiago	API REST + GTFS	Mensual
Áreas verdes	Sentinel-2 (GEE)	Cálculo NDVI	Quincenal
POIs comerciales	OpenStreetMap	Overpass API	Semanal
Demografía	Censo 2017 (INE)	Agregación por manzana	Estático
Delitos	Subsecretaría Prevención	Geocodificación + KDE	Trimestral
Calidad del aire	SINCA	Interpolación IDW	Horaria
Permisos construcción	DOM municipales	Scraping + análisis	Mensual

3.4. Indicadores de Éxito

- MAPE ¡12 % en predicción de precios
- R^2 ¿0.85 en validación cruzada espacial
- Tiempo de respuesta ¡2 segundos por consulta

- Cobertura ¿90 % del Gran Santiago

4. Propuesta de Proyecto 2: Plataforma de Inteligencia Locacional para Retail

4.1. Descripción General

Sistema de análisis espacial avanzado para determinar ubicaciones óptimas de nuevos locales comerciales, prediciendo su rendimiento basado en variables demográficas, de competencia y accesibilidad.

4.2. Componentes del Sistema

```python
import geopandas as gpd
import networkx as nx
from scipy.spatial import Voronoi
from sklearn.cluster import DBSCAN
import h3

class RetailLocationIntelligence:
    def __init__(self, city_bounds):
        self.city = city_bounds
        self.competitors = gpd.GeoDataFrame()
        self.demographics = gpd.GeoDataFrame()
        self.foot_traffic = {}

    def analyze_market_saturation(self, business_type):
        """Analiza saturaci n del mercado por tipo de negocio
           """
        # Obtener competidores
        competitors = self.get_competitors_osm(business_type)

        # Crear  reas  de influencia (Voronoi)
        vor = Voronoi(competitors[['x', 'y']])
        market_areas = self.voronoi_to_geodataframe(vor)

        # Calcular  ndice  de saturaci n
        for area in market_areas:
            population = self.get_population_in_area(area)
            n_competitors = len(competitors.within(area))
            area['saturation_index'] = n_competitors /
               population * 1000

        return market_areas

    def calculate_accessibility_score(self, location):
        """Calcula score de accesibilidad multimodal"""
        scores = {}

        # 1. Accesibilidad peatonal (is cronas)
        walking_iso = self.calculate_isochrone(
            location, mode='walk', time=10
        )
```

```
        scores['walking'] = self.population_in_polygon(walking_iso)

        # 2. Accesibilidad en auto
        driving_iso = self.calculate_isochrone(
            location, mode='drive', time=15
        )
        scores['driving'] = self.population_in_polygon(driving_iso)

        # 3. Transporte p blico
        transit_iso = self.calculate_isochrone(
            location, mode='transit', time=20
        )
        scores['transit'] = self.population_in_polygon(transit_iso)

        # 4. Estacionamientos cercanos
        parking = self.count_parking_spots(location, radius=300)
        scores['parking'] = parking

        # Score ponderado
        weights = {'walking': 0.4, 'driving': 0.2,
                   'transit': 0.3, 'parking': 0.1}
        total_score = sum(scores[k] * weights[k] for k in scores)

        return total_score, scores

    def predict_foot_traffic(self, location, datetime):
        """Predice tr fico peatonal usando datos m viles"""
        # Usar hex gonos H3 para agregaci n
        h3_index = h3.geo_to_h3(
            location.y, location.x, resolution=9
        )

        # Features temporales
        hour = datetime.hour
        day_of_week = datetime.dayofweek
        is_weekend = day_of_week >= 5

        # Features espaciales
        nearby_pois = self.count_pois_by_category(location, radius=200)

        # Modelo de predicci n (pre-entrenado)
        features = [hour, day_of_week, is_weekend] + list(nearby_pois.values())
        predicted_traffic = self.traffic_model.predict([features])[0]

        return predicted_traffic

    def optimize_location_portfolio(self, n_locations, constraints):
        """Optimiza portfolio de ubicaciones"""
        from scipy.optimize import differential_evolution
```

```
88
89          def objective(locations):
90              # Reshape a coordenadas
91              locs = locations.reshape(n_locations, 2)
92
93              total_coverage = 0
94              overlap_penalty = 0
95
96              for i, loc in enumerate(locs):
97                  # Cobertura de mercado
98                  coverage = self.calculate_market_coverage(loc)
99                  total_coverage += coverage
100
101                 # Penalizaci n por solapamiento
102                 for j in range(i+1, len(locs)):
103                     distance = np.linalg.norm(loc - locs[j])
104                     if distance < constraints['min_distance']:
105                         overlap_penalty += 1000 / distance
106
107             return -(total_coverage - overlap_penalty)
108
109         # Optimizaci n
110         bounds = [(self.city.bounds[0], self.city.bounds[2])] *
                (n_locations * 2)
111         result = differential_evolution(objective, bounds)
112
113         optimal_locations = result.x.reshape(n_locations, 2)
114         return self.create_location_report(optimal_locations)
115
116     def generate_opportunity_map(self):
117         """Genera mapa de oportunidades comerciales"""
118         # Crear grid de an lisis
119         grid = self.create_analysis_grid(resolution=100)
120
121         for cell in grid:
122             centroid = cell.centroid
123
124             # Calcular m tricas
125             cell['population_density'] = self.
                    get_population_density(centroid)
126             cell['competition_index'] = self.
                    calculate_competition(centroid)
127             cell['accessibility'] = self.
                    calculate_accessibility_score(centroid)[0]
128             cell['income_level'] = self.get_average_income(
                    centroid)
129
130             # Score de oportunidad
131             cell['opportunity_score'] = (
132                 cell['population_density'] * 0.3 +
133                 (1 - cell['competition_index']) * 0.3 +
134                 cell['accessibility'] * 0.2 +
135                 cell['income_level'] * 0.2
136             )
137
138         return self.visualize_opportunity_grid(grid)
```

Listing 2: Motor de análisis locacional

4.3. Análisis de Competencia Espacial

```
def spatial_competition_analysis(self, business_location,
    competitor_locations):
    """An lisis de competencia usando modelos de gravitaci n
       """
    import numpy as np
    from scipy.spatial.distance import cdist

    # Modelo de Huff para probabilidad de visita
    def huff_model(target, competitors, alpha=1, beta=2):
        # Atractivo de cada tienda (tama o, calidad, etc.)
        attractiveness = np.array([store['rating'] * store['
            size']
                                   for store in competitors])

        # Distancias
        distances = cdist([target], competitor_locations)[0]

        # Utilidad de cada tienda
        utilities = attractiveness / (distances ** beta)

        # Probabilidad de visitar tienda objetivo
        prob_target = utilities[0] / np.sum(utilities)

        return prob_target

    # An lisis de canibalizaci n
    def cannibalization_risk(new_location, existing_locations):
        overlap_areas = []
        for existing in existing_locations:
            #  rea de influencia compartida
            overlap = self.calculate_overlap_area(new_location,
                existing)
            overlap_areas.append(overlap)

        total_cannibalization = sum(overlap_areas)
        return total_cannibalization / self.total_market_area

    return {
        'market_share': huff_model(business_location,
            competitor_locations),
        'cannibalization': cannibalization_risk(
            business_location, existing_stores),
        'competitive_advantage': self.
            calculate_competitive_metrics(business_location)
    }
```

Listing 3: Análisis avanzado de competencia

5. Propuesta de Proyecto 3: Monitor de Desarrollo Urbano y Plusvalía

5.1. Descripción General

Sistema de monitoreo continuo que detecta cambios urbanos usando imágenes satelitales y predice su impacto en la plusvalía inmobiliaria, generando alertas tempranas de oportunidades de inversión.

5.2. Tecnologías Clave

- **Detección de cambios:** Análisis multitemporal con Sentinel-2
- **Deep Learning:** CNN para clasificación de uso de suelo
- **Series temporales:** Prophet para proyección de plusvalía
- **Visualización:** Dashboard interactivo con Streamlit

5.3. Pipeline de Procesamiento

```
import ee
import tensorflow as tf
from prophet import Prophet
import streamlit as st

class UrbanDevelopmentMonitor:
    def __init__(self):
        ee.Initialize()
        self.model = self.load_pretrained_model()

    def detect_urban_changes(self, aoi, date_start, date_end):
        """Detecta cambios urbanos usando Sentinel-2"""
        # Obtener im genes
        col_before = ee.ImageCollection('COPERNICUS/S2_SR')\
            .filterBounds(aoi)\
            .filterDate(date_start, ee.Date(date_start).advance
               (3, 'month'))\
            .median()

        col_after = ee.ImageCollection('COPERNICUS/S2_SR')\
            .filterBounds(aoi)\
            .filterDate(date_end, ee.Date(date_end).advance(3,
               'month'))\
            .median()

        #  ndices  espectrales
        ndbi_before = self.calculate_ndbi(col_before)
        ndbi_after = self.calculate_ndbi(col_after)

        # Detecci n de cambios
        change = ndbi_after.subtract(ndbi_before)

        # Clasificar tipos de cambio
        new_construction = change.gt(0.15)
```

```
        demolition = change.lt(-0.15)

        return {
            'new_construction': new_construction,
            'demolition': demolition,
            'change_magnitude': change
        }

    def classify_development_type(self, change_area):
        """Clasifica tipo de desarrollo usando CNN"""
        # Extraer chip de imagen
        image_chip = self.extract_image_chip(change_area, size
            =224)

        # Predicci n con modelo pre-entrenado
        predictions = self.model.predict(image_chip)

        classes = ['residential', 'commercial', 'industrial',
                   'infrastructure', 'park', 'mixed_use']

        return {
            'type': classes[np.argmax(predictions)],
            'confidence': float(np.max(predictions))
        }

    def predict_plusvalue_impact(self, location,
        development_type):
        """Predice impacto en plusval a"""
        # Datos hist ricos de proyectos similares
        historical_data = self.get_historical_impacts(
            development_type,
            radius=2000
        )

        # Preparar serie temporal
        df = pd.DataFrame({
            'ds': historical_data['date'],
            'y': historical_data['price_change_pct']
        })

        # Agregar regresores
        df['distance'] = self.calculate_distance_to_development
            (location)
        df['size'] = self.get_development_size()
        df['type_commercial'] = int(development_type == '
            commercial')
        df['type_infrastructure'] = int(development_type == '
            infrastructure')

        # Modelo Prophet con regresores
        model = Prophet(
            changepoint_prior_scale=0.05,
            seasonality_mode='additive'
        )
        model.add_regressor('distance')
        model.add_regressor('size')
        model.add_regressor('type_commercial')
        model.add_regressor('type_infrastructure')
```

```
86
87          model.fit(df)
88
89          # Predicci n a 24 meses
90          future = model.make_future_dataframe(periods=24, freq='
                M')
91          future['distance'] = df['distance'].iloc[-1]
92          future['size'] = df['size'].iloc[-1]
93          future['type_commercial'] = df['type_commercial'].iloc
                [-1]
94          future['type_infrastructure'] = df['type_infrastructure
                '].iloc[-1]
95
96          forecast = model.predict(future)
97
98          return {
99              'expected_appreciation': forecast['yhat'].iloc[-1],
100             'confidence_interval': (forecast['yhat_lower'].iloc
                    [-1],
101                                     forecast['yhat_upper'].iloc
                                            [-1]),
102             'time_to_peak': self.find_peak_time(forecast)
103         }
104
105     def create_investment_alert(self, opportunity):
106         """Genera alerta de oportunidad de inversi n"""
107         score = self.calculate_opportunity_score(opportunity)
108
109         if score > 0.8:
110             alert = {
111                 'level': 'HIGH',
112                 'message': f"Oportunidad de inversi n
                        detectada en {opportunity['location']}",
113                 'expected_roi': opportunity['
                        expected_appreciation'],
114                 'risk_level': self.assess_risk(opportunity),
115                 'recommended_action': self.
                        generate_recommendation(opportunity)
116             }
117
118             # Enviar notificaci n
119             self.send_alert(alert)
120
121         return alert
```

Listing 4: Detección de cambios urbanos y predicción de plusvalía

6. Recomendaciones de Implementación

6.1. Plan de Desarrollo Sugerido

Semana	Fase	Actividades	Entregable
1-2	Fundamentos	Aprender GeoPandas, PostGIS básico	Notebook tutorial
3-4	Datos	Recolección y limpieza de datos inmobiliarios	Dataset limpio
5-6	Análisis	Exploración espacial, correlaciones	EDA completo
7-8	Modelado	Desarrollo de modelos ML	Modelos entrenados
9-10	Validación	Testing y ajuste de parámetros	Métricas finales
11-12	Visualización	Dashboard interactivo	App Streamlit
13-14	Documentación	Manual técnico y de usuario	Documentación
15	Presentación	Preparación presentación final	Demo en vivo

6.2. Stack Tecnológico Recomendado

TECNOLOGÍAS ESENCIALES

- **Python:** GeoPandas, Shapely, Rasterio, Folium
- **Base de datos:** PostgreSQL + PostGIS
- **Machine Learning:** Scikit-learn, XGBoost, Prophet
- **Visualización:** Streamlit, Plotly, Kepler.gl
- **Cloud:** Google Earth Engine para procesamiento satelital
- **APIs:** OpenStreetMap Overpass, Transporte público GTFS

6.3. Recursos de Aprendizaje Prioritarios

1. **GeoPandas fundamentals:**
   - Curso: "Spatial Analysis with Python"(2 semanas)
   - Documentación oficial de GeoPandas
2. **Machine Learning Espacial:**
   - Paper: "Machine Learning for Spatial Environmental Data"
   - Tutorial: Spatial Cross-Validation techniques
3. **Mercado Inmobiliario:**
   - Informes trimestrales CCHC
   - Portal de datos abiertos del SII

7. Métricas de Evaluación del Proyecto

7.1. Criterios Técnicos (50 %)

- Correcta implementación de análisis espacial
- Calidad del código y documentación
- Performance de modelos ML (MAPE, R^2)
- Manejo eficiente de datos geoespaciales

7.2. Criterios de Negocio (30 %)

- Viabilidad comercial de la solución
- Claridad del modelo de negocio
- Potencial de escalabilidad
- Diferenciación de competencia

7.3. Criterios de Innovación (20 %)

- Originalidad del enfoque
- Uso creativo de datos alternativos
- Integración de múltiples fuentes
- Potencial de publicación académica

8. Riesgos y Mitigaciones

Riesgo	**Probabilidad**	**Impacto**	**Mitigación**
Disponibilidad de datos inmobiliarios	Alta	Alto	Web scraping + múltiples fuentes
Complejidad de GeoPandas	Media	Medio	Tutoriales + office hours
Overfitting en modelos	Media	Alto	Cross-validation espacial rigurosa
Performance con big data	Baja	Medio	Uso de Dask + optimización

9. Conclusiones y Siguientes Pasos

9.1. Fortalezas Identificadas

- **Base técnica sólida:** Python y SQL facilitan la curva de aprendizaje
- **Visión clara:** Interés específico en inmobiliario reduce ambigüedad

- **Mercado atractivo:** Sector inmobiliario chileno en crecimiento
- **Aplicabilidad inmediata:** Proyectos con potencial comercial real

9.2. Recomendación Final

RECOMENDACIÓN DEL PROFESOR

Byron debe enfocarse en el Proyecto 1: Sistema Inteligente de Valoración Inmobiliaria
Razones:

1. Alinea perfectamente con sus intereses en negocios y plusvalía
2. Aprovecha su fortaleza en Python para ML
3. Tiene aplicación comercial inmediata
4. Permite aprendizaje gradual de conceptos GIS
5. Potencial para convertirse en startup

Primeros pasos sugeridos:

1. Instalar ambiente: conda install geopandas folium scikit-learn
2. Tutorial GeoPandas: 2-3 días de práctica intensiva
3. Comenzar con dataset de 100 propiedades para prototipo
4. Implementar modelo básico con 5 variables urbanas
5. Iterar agregando complejidad gradualmente

Meta: Tener MVP funcional en 4 semanas

9.3. Cronograma Inicial Detallado

Semana 1: Inmersión en Geoinformática

- Lunes-Martes: Tutorial completo de GeoPandas
- Miércoles-Jueves: Ejercicios con datos de Santiago
- Viernes: Configurar PostgreSQL + PostGIS

Semana 2: Recolección de Datos

- Lunes-Martes: Web scraping Portal Inmobiliario
- Miércoles: Geocodificación de direcciones
- Jueves-Viernes: Descarga datos OSM y censo

Semana 3: Análisis Exploratorio

- Lunes-Martes: Cálculo de variables urbanas
- Miércoles-Jueves: Visualizaciones y correlaciones
- Viernes: Identificar patrones espaciales

Semana 4: Primer Modelo

- Lunes-Martes: Implementar regresión baseline
- Miércoles-Jueves: Agregar XGBoost con tuning
- Viernes: Evaluación y presentación de avance

10. Material de Apoyo Adicional

10.1. Código de Inicio Rápido

```
# Instalar librer as necesarias
# pip install geopandas folium scikit-learn beautifulsoup4
    requests

import geopandas as gpd
import pandas as pd
import folium
from shapely.geometry import Point
import requests
from bs4 import BeautifulSoup

# 1. Crear primer GeoDataFrame
data = {
    'direccion': ['Providencia 123', 'Las Condes 456', '  uoa
        789'],
    'precio': [150000000, 280000000, 95000000],
    'm2': [65, 110, 45],
    'lat': [-33.4489, -33.4089, -33.4589],
    'lon': [-70.6693, -70.5693, -70.6093]
}

df = pd.DataFrame(data)
geometry = [Point(xy) for xy in zip(df.lon, df.lat)]
gdf = gpd.GeoDataFrame(df, geometry=geometry, crs='EPSG:4326')

# 2. Calcular precio por m2
gdf['precio_m2'] = gdf['precio'] / gdf['m2']

# 3. Crear buffer de 500m alrededor de cada propiedad
gdf_utm = gdf.to_crs('EPSG:32719')  # Proyectar a UTM para
    Chile
gdf_utm['buffer'] = gdf_utm.geometry.buffer(500)

# 4. Visualizar en mapa interactivo
m = folium.Map(location=[-33.45, -70.65], zoom_start=12)

for idx, row in gdf.iterrows():
    folium.CircleMarker(
```

```
        [row.geometry.y, row.geometry.x],
        radius=5,
        popup=f"Precio: ${row['precio']:,.0f}<br>m  : {row['m2
            ']}<br>$/m  : ${row['precio_m2']:,.0f}",
        color='red',
        fill=True
    ).add_to(m)

m.save('mi_primer_mapa_inmobiliario.html')
print("Mapa creado exitosamente!")

# 5. An lisis espacial b sico
from sklearn.neighbors import NearestNeighbors

# Encontrar 3 propiedades m s cercanas a cada una
coords = gdf_utm[['geometry']].apply(lambda x: [x.geometry.x, x
    .geometry.y], axis=1).tolist()
nbrs = NearestNeighbors(n_neighbors=3, algorithm='ball_tree').
    fit(coords)
distances, indices = nbrs.kneighbors(coords)

print("\nDistancias a vecinos m s cercanos (metros):")
for i, dist in enumerate(distances):
    print(f"Propiedad {i}: {dist[1]:.0f}m y {dist[2]:.0f}m")

# Este es tu punto de partida, Byron.  Ahora  a construir algo
    incre ble!
```

Listing 5: Script inicial para Byron

10.2. Enlaces Útiles

- **Datos Santiago:** https://datos.gob.cl/dataset?q=santiago
- **IDE Chile:** http://www.ide.cl/
- **GeoPandas Docs:** https://geopandas.org/
- **Ejemplos ML Espacial:** https://github.com/giswqs/geospatial-machine-learning

Dr. Francisco Parra O.
Profesor Curso Geoinformática
Universidad de Santiago de Chile
francisco.parra.o@usach.cl

¡Éxito en tu proyecto, Byron!
Agosto 2025