Predicción del precio de viviendas cubanas

Leonardo Artiles Montero LEO16AM@GMAIL.COM
Grupo C112

Alex Samuel Bas Beovides ABASBEOVIDES@GMAIL.COM
Grupo C112

Ariel González Gómez LENIN46ARIEL@GMAIL.COM
Grupo C112

Edián Broche Castro EDIANBC@GMAIL.COM
Grupo C112

Tutor(es):
Dr. Yudivian Almeida Cruz, *MATCOM*
Dra. Suilan Estevez-Velarde, *MATCOM*
Dr. Alejandro Piad Morffis, *MATCOM*

Resumen

La predicción de precios de viviendas cubanas es importante para que tanto vendedores como compradores tomen decisiones informadas. En este artículo se extrajeron datos del mayor sitio de compra y venta de casas de Cuba para formar un conjunto de datos. Se entrenaron dos modelos de Aprendizaje de Máquina, uno con Refuerzo Extremo del Gradiente (XGBoost) y otro con Redes Neuronales Artificiales, para realizar la predicción del precio de las propiedades basado en sus características. Se obtuvieron resultados más satisfactorios con el modelo de redes neuronales artificiales y a partir del cual se desarrolló una aplicación web para que quien desee conocer el precio en el mercado de un domicilio obtenga un resultado aproximado.

Abstract

Forecasting Cuban house prices is important for both sellers and buyers to make informed decisions. In this paper, data was extracted from the largest house buying and selling site in Cuba to form a dataset. Two Machine Learning models were trained, one with Extreme Gradient Boosting (XGBoost) and another with Artificial Neural Networks, to forecast the price of the properties based on their features. More satisfactory results were obtained with the neural network model and from which a web application was developed so that whoever wants to know the market price of a house can obtain an approximate result.

Palabras Clave: Regresión, redes neuronales artificiales, árboles de decisión, precio de viviendas.
Tema: Tema, Subtema.

1. Introducción

A la hora de vender una vivienda encontrar el precio indicado conlleva a mayor competitividad en el mercado inmobiliario y una retribución monetaria justa para el propietario, por tanto es un problema común y de gran relevancia. El valor de la moneda en Cuba es bastante fluctuante debido a la inflación y esto afecta directamente el precio de las viviendas[1], asi mismo como la crisis migratoria. Es por ello que los intentos anteriores de crear modelos para la predicción automática del precio de domicilios quedan obsoletos con el tiempo. El objetivo de este trabajo es crear un modelo de Aprendizaje Automatico actualizado para realizar dicha predicción.

Existen varias aproximaciones a este problema, entre ellas se encuentra la regresión hedónica [2], la cual se basa en la descomposición de un bien económico en sus características más importantes y en el análisis de la contribución al valor agregado de cada una de tales características. Se destaca el uso de árboles de decisión [3] los cuales crean diagramas de construcciones lógicas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema. Son usados de igual manera los bosques aleatorios que son una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos. Se usan el Refuerzo Extremo del Gradiente [4] y las Redes Neuronales Artificiales [5] las cuales son grupo interconectado de nodos similar a la vasta red de neuronas en un cerebro biológico que realizan su aprendizaje mediante el descenso del gradiente. Estos dos últimos algoritmos fueron los seleccionados en esta investigación.

2. Conjunto de datos

La mayor web de comercio actualmente en Cuba revolico.com es también la mayor web de ventas de inmuebles, es por esto que se decide usar sus datos para el entrenamiento de nuestra solución. Esta web guarda las últimos 10000 anuncios publicados, en formato de texto, más los indicadores del precio, la provincia y la localidad en la que se encuentra ubicada la vivienda. Un anuncio común en revolico puede ser el siguiente:

> "Se vende casa en óptimas condiciones con todo adentro es llegar y vivir La Casa Consta de: Portal Cerrado, Pasillos Lateral con Acceso y Puerta con Candado, Sala, 2 Cuartos con Closet, Baño Nuevo Azulejeado entre los dos cuartos, Comedor, Cocina Completamente Azulejeada, Terraza cerrada y de Placa con Lavaderos Azulejeados, Patio de Cemento y de Tierra, Cuartico de Desahogo, Placa Libre con Escalera de Cemento para fácil acceso con dos Tanques Enormes Nuevos y Tendederas. Toda la casa tiene Ventanas y Puertas de Hierro y Cristales, tiene Lámparas de la Tienda en toda la casa con sus Bombillos LED, Las Puertas interiores son de las buenas de la tienda, Toda la casa cuenta con Sistemas de Agua Fría y Caliente con Presurizador y Calentador Eléctrico Nuevo.Con la compra se deja Refrigerador Haier(No son los q se ven en la foto), Cocina de Gas de 4 Hornillas con Horno(Contrato de Gas) Microwave, Arrocera, Batidora y muchas cosas mas."

Se realizó web scrapping [6] del sitio web, el cual duró aproximadamente 49 horas. Y se logro descargar 6500 anuncios, descartando a los repetidos y a los que tenían precio menor que 1000 dólares o mayor que 5000000 quedaron 6296.

2.1 Extracción de Enlaces

Se utilizó la biblioteca de Python Selenium [7], que permite trabajar con el código de la pagina web de forma dinámica. Extrayendo del código HTML de la página los elementos, se pudo acceder al precio, localización, y enlace a la página con la descripción de cada anuncio. Se realizó automáticamente la transformación pertinente entre las divisas usando el cambio existente en el momento para facilitar el trabajo del entrenamiento posteriormente. Finalmente se guardaron los datos capturados en un .csv, para su ulterior trato. El tiempo de ejecución de este segmento del código fue de una hora aproximadamente.

2.2 Extracción de descripciones

Empleando la misma metodología del apartado anterior, se recorrieron los enlaces previamente descargados, y se extrajeron las descripciones contenidas en cada un. Se guardaron los nuevos datos adquiridos cada 50 iteraciones, para prevenir en caso de intervenciones relacionadas con la conexión o el fluido eléctrico. El tiempo de ejecución de este segmento del código, sumado entre varias sesiones, fue de 49 horas aproximadamente. Se presentaron varios problemas mientras se realizaba el web scrapping debido a la seguridad del sitio para proteger los datos:

Debido a que el sitio esta protegido con captchas, existió la necesidad de recargar el driver web cada cierta cantidad de descargas, esto solucionó dicho problema. El sitio cambia las clases de los objetos en el codigo HTML regularmente para su protección, obligando a que se tuviera que cambiar el código del scrapper. Finalmente se obtuvo un conjunto de datos de 6mb en formato csv. En el archivo final del conjunto de datos se tenian cuatro columnas: precio de la vivienda, provincia, municipio y la descripcion en texto .

3. Extracción de caractrísticas

Usando la idea principal de la regresión hedónica de que las distintas características de la vivienda afectan por separado el precio final se decidió extraer una por una las caractrísticas consideradas importantes. Se llegó a la conclusión de que dichas catacterísticas podían ser clasificadas en dos tipos básicos, booleanos o numéricos. Las características de tipo booleano son las que pueden ser verdaderas o falsas, por ejemplo el hecho de que la vivienda sea un apartamento o posea gas. Las características numéricas indican una cantidad, como puede ser el número de cuartos o de tanques de agua.

Para cada característica se usaron varios tokens que podían representarla en el texto, por ejemplo las habitaciones pueden ser representadas por el token "habitación" o "cuarto". Para ver si alguna característica booleana es positiva se buscó si alguno de sus tokens aparecían en el texto y ninguna de las dos palabras precedentes indican una negación, o sea ninguna es la palabra "no". Para las características numéricas se buscó su respectivo token y en caso de que alguna de las dos palabras precedentes fueran un número se usó este como la cantidad de esa característica. Uno de los principales problemas encontrados fue que en revolico.com a la hora de poner la cantidad de cuartos de una vivienda muchos usuarios escriben "3/4" para indicar que esta posee 3 habitaciones, para resolver esto se creó una función que chequeaba dicho caso.

3.1 Corrección de errores ortográficos

Son comunes los errores ortográficos o abreviaciones en revolico.com. Cuando se buscan los tokens de las características resulta un problema. Para ello se utilizó la distancia de Levenshtein [8] como métrica de la diferencia entre dos tokens. La distancia de Levenshtein indica la mínima cantidad de operaciones de edición que se le deben realizar a una palabra para convertirla en otra, donde una operación puede ser cambiar una letra por otra, añadir un letra o quitarla. Se consideró que dos tokens son el mismo si ambos tienen distancia menor que 2 y su tamaño es mayor que 5. En el caso que el tamaño sea menor o igual que 5 se considera que

	Característica	Tipo
1	Apartamento	Booleano
2	Gas	Booleano
4	Amueblado	Booleano
5	Piscina	Booleano
6	Patio	Booleano
7	Nauta Hogar	Booleano
8	Cisterna	Booleano
9	Finca	Booleano
10	Portal	Booleano
11	Techo de placa	Booleano
12	Techo de tejas	Booleano
13	Cercano a un Hospital	Booleano
14	Cercano a una Escuela	Booleano
15	Cuartos	Numérico
16	Baños	Numérico
17	Cocinas	Numérico
18	Salas	Numérico
19	Plantas	Numérico
20	Comedores	Numérico
21	Desahogos	Numérico
22	Terrazas	Numérico
23	Carros	Numérico
24	Splits	Numérico
25	Garajes	Numérico
26	Tanques	Numérico
27	Refrigeradores	Numérico

Figura 1: Características extraidas de cada texto y su respectivo tipo.

las palabras son iguales si su distancia es igual a cero.

3.2 Implicaciones lógicas

La ausencia de datos es uno de los principales problemas para la robustez del modelo. Para completar los datos faltantes uno de los métodos utilizados es el uso de implicaciones lógicas dados los datos ya conocidos. Ejemplo de esto es que si una casa es una finca entonces posee patio. Se dividieron las implicaciones en tres tipos, el primer tipo expresa que si una característica A es verdadera implica que va a existir la característica B. El segundo tipo indica que si la característica A es verdadera entonces la característica B es falsa. El tercer tipo es la implicación de que si una característica no es mencionada entonces no va a existir.

3.3 Completamiento de datos faltantes

Los datos faltantes que no pudieron ser completados por las implicaciones lógicas se completaron usando el promedio de los datos existentes.

3.4 Representación de datos categóricos

Los datos categóricos (provincias y municipios) fueron representados mediante el one-hot encoding [9] de dichas características.

4. Modelo XGBoost

Extreme Gradient Boosting (XGBoost), es una biblioteca distribuida y escalable de aprendizaje automático de árboles de decisión con refuerzo de gradiente (ADRG). Proporciona refuerzo de árboles en paralelo y es la biblioteca de aprendizaje automático líder para problemas de regresión, clasificación y ranking [10].

Para comprender XGBoost, es vital comprender primero los conceptos y algoritmos de aprendizaje automático en los que se basa XGBoost: aprendizaje automático supervisado, árboles de decisión, ensemble learning y refuerzo de gradiente.

El aprendizaje automático supervisado usa algoritmos para entrenar un modelo con el objetivo de encontrar patrones en un conjunto de datos con etiquetas y características. Luego usa el modelo entrenado para predecir las etiquetas en las características de un nuevo conjunto de datos [11].

Los árboles de decisión crean un modelo que predice la etiqueta mediante la evaluación de un árbol de consultas de características de verdadero/falso de tipo if-then-else, y estimando la cantidad mínima de consultas para evaluar la probabilidad de tomar una decisión correcta. Los árboles de decisión se pueden usar para la clasificación para predecir una categoría o la regresión para predecir un valor numérico contínuo [12].

Un árbol de decisión de refuerzo de gradiente es un algoritmo de ensemble learning de árboles de decisión similar al de bosque aleatorio, para clasificación y regresión. Los algoritmos de ensemble learning combinan múltiples algoritmos de aprendizaje automático para obtener un mejor modelo [13].

Tanto los bosques aleatorios como los árboles de decisión con refuerzo de gradiente construyen un modelo que consta de múltiples árboles de decisión. La diferencia está en cómo se construyen y combinan los árboles.

Los bosques aleatorios utilizan una técnica llamada bagging [14] para construir árboles de decisión completos en paralelo a partir de muestras aleatorias de arranque del conjunto de datos. La predicción final es un promedio de todas las predicciones del árbol de decisiones.

El término "refuerzo de gradiente" proviene de la idea de "reforzar" o mejorar un solo modelo débil combinándolo con una serie de otros modelos débiles para generar un modelo fuerte colectivo. El refuerzo de gradiente es una extensión del refuerzo donde el proceso de generación aditiva de modelos débiles se formaliza como un algoritmo de descenso de gradiente sobre una función objetivo. El refuerzo de gradiente establece resultados específicos para el próximo modelo en un esfuerzo por minimizar los errores. Los resultados previstos para cada caso se basan en el gradiente del error (de ahí el nombre de refuerzo del gradiente) con respecto a la predicción.

Los ADRG entrenan iterativamente un conjunto de árboles de decisión poco profundos, y cada iteración usa los residuos de error del modelo anterior para ajustar el siguiente modelo. La predicción final es una suma

ponderada de todas las predicciones del árbol. El "bagging" de bosques aleatorios minimiza la varianza y el sobreajuste (overfitting), mientras que el "refuerzo" de ADRG minimiza el sesgo (bias) y el desajuste (underfitting).

XGBoost es una implementación escalable y muy precisa de refuerzo de gradiente que supera los límites de la potencia informática para los algoritmos de árbol reforzado, y se crea en gran medida para potenciar el rendimiento del modelo de aprendizaje automático y la velocidad computacional. Con XGBoost, los árboles se construyen en paralelo, en lugar de secuencialmente como ADRG. Sigue una estrategia por niveles, escaneando valores de gradiente y usando estas sumas parciales para evaluar la calidad de las divisiones en cada división posible en el conjunto de entrenamiento.

Todas las características y beneficios que posee XGBoost lo convierten en un candidato perfecto para su utilización en el proyecto que nos concierne, al tratarse de un problema relativamente simple de regresión en datos tabulares para predecir un valor numérico continuo (el precio) basado en valores numéricos de entrada (features).

Otro concepto fundamental de los modelos de Aprendizaje de Máquinas son los hiperparámetros, los cuales son ciertos valores o pesos que determinan el proceso de aprendizaje. En los modelos basados en árboles, los hiperparámetros incluyen aspectos como la profundidad máxima del árbol, la cantidad de árboles a entrenar, la cantidad de variables a considerar al construir cada árbol, la cantidad mínima de muestras en una hoja y la fracción de observaciones utilizadas para construir un árbol. En el fragmento de código siguiente se observan los hiperparámetros utilizados en el entrenamiento de nuestro modelo:

```
XGBRegressor(colsample_bytree=1,
    gamma=0,
    learning_rate=0.08,
    max_depth=50,
    min_child_weight=5,
    n_estimators=10000,
    subsample=0.06)
```

Figura 2: Hiperparámetros de XGBoost.

5. Modelo de Keras

Otro enfoque usado son las redes neuronales artificiales para la regresión del precio de las casas.

Las redes neuronales artificiales son un modelo computacional que consiste en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida [15].

Cada neurona está conectada con otras a través de unos enlaces. En estos enlaces el valor de salida de la neurona anterior es multiplicado por un valor de peso. Estos pesos en los enlaces pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Del mismo modo, a la salida de la neurona, puede existir una función limitadora o umbral, que modifica el valor resultado o impone un límite que no se debe sobrepasar antes de propagarse a otra neurona. Esta función se conoce como función de activación.

Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional. Para realizar este aprendizaje automático, normalmente, se intenta minimizar una función de pérdida que evalúa la red en su total. Los valores de los pesos de las neuronas se van actualizando buscando reducir el valor de la función de pérdida. Este proceso se realiza mediante la propagación hacia atrás.

Se realizó el experimento usando Keras la cual es una biblioteca de Redes Neuronales de código abierto escrita en Python. Contiene varias implementaciones de los bloques constructivos de las redes neuronales como por ejemplo las capas, funciones de pérdida, funciones de activación y optimizadores matemáticos. Utilizando esta biblioteca se creó una red de 5 capas con la siguiente arquitectura.

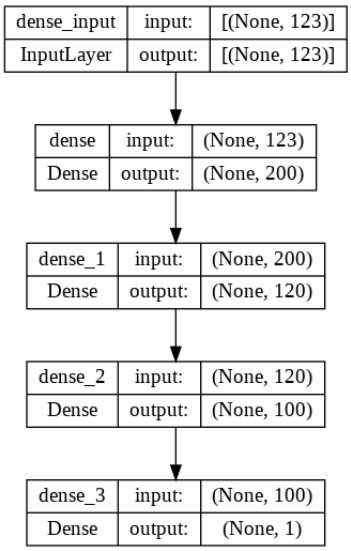

Figura 3: Arquitectura del modelo de Keras.

6. Página web

Se construyó una simple página web para uso público en la cual un usuario puede insertar la descripción de una vivienda en formato textual (como si se tratase de un anuncio de revolico.com) además de la provincia y el municipio en que se encuentra, y el sitio se encargará de dar un precio aproximado de dicha vivienda.

Se utilizó Flask, el cual es un framework de Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. En el back-end de Python se accede al modelo de Redes Neuronales

Artificiales de Keras y se usa este para realizar la predicción.

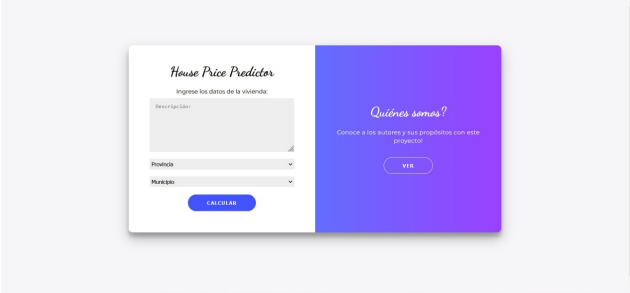

Figura 4: Página de inicio.

Figura 5: Resultados de predicción.

7. Resultados

El web-scrapping del sitio de ventas revolico.com resultó satisfactorio, a pesar de los problemas encontrados durante el proceso. Así mismo fue satisfactoria la extracción de las características(features) de cada anuncio.

Comúnmente se usan tres métricas para evaluar y reportar el desempeño de un modelo de regresión, estas son: el error absoluto medio (MAE), el error cuadrático medio (MSE) y la raíz del error cuadrático medio (RMSE).

El error absoluto medio (MAE) se define como la media del valor absoluto de los errores:

$$\sum_{i=1}^{N} \frac{|x_i - y_i|}{N}$$

El error cuadrático medio (MSE) se define como la media de los errores al cuadrado. Es más popular que el MAE porque el enfoque se orienta más hacia grandes errores. Esto se debe a que el término al cuadrado aumenta exponencialmente los errores más grandes en comparación con los más pequeños. Su fórmula es:

$$\sum_{i=1}^{N} \frac{(x_i - y_i)^2}{N}$$

La raíz cuadrada del error cuadrático medio (RMSE) se define como la raíz cuadrada de la anterior medida. Esta es una de las métricas más populares de las métricas de evaluación porque es interpretable en las mismas unidades que el vector de respuesta, haciendo fácil de correlacionar con la información. Su fórmula es:

$$\sum_{i=1}^{N} \sqrt{\frac{(x_i - y_i)^2}{N}}$$

Asismismo se encuentra el error porcentual absoluto medio, en inglés Mean Absolute Percentage Error (MAPE), el cual se usa comúnmente como una función de pérdida para problemas de regresión y en la evaluación de modelos, debido a su interpretación muy intuitiva en términos de error relativo, y por tanto fue la utilizada en nuestro proyecto. Se define como:

$$\frac{100\,\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

A continuación se muestra un gráfico del MAPE en función de las épocas de entrenamiento del modelo de Keras:

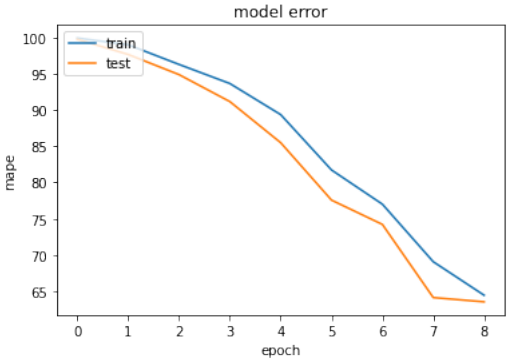

Figura 6: Comportamiento del error del modelo.

Se alcanzó un MAPE en la predicción de 63.5603 % usando el modelo de Keras.

A diferencia de lo esperado, el modelo de XGBoost resultó tener un resultado muy pobre en comparación con el modelo de redes neuronales, alcanzando un MAPE de aproximadamente 123 %. Este hecho se podría deber a una combinación poco óptima de los hiperparámetros (tuning) y a la escasez de datos de entrenamiento y su depurado.

8. Discusión

Entre los principales problemas encontrados y que afectaron directamente el proceso de entrenamiento y la precisión del modelo están:

1. Poca cantidad de datos de entrenamiento.
2. Varición de los precios en un corto período de tiempo e inflación de la moneda.
3. Exsitencia de anuncios repetidos lexicográficamente distintos.
4. Existencia de anuncios falsos o con precios irreales.
5. Errores en la extracción de características.
6. Mala representación de los datos categóricos.

Se observó también que a la hora de predecir precios el modelo tiende a ofrecer un valor menor al real. Se teoriza de que esto puede deberse a la baja que está

sufriendo el mercado inmobiliario actual. Debido a la crisis migratoria, muchas personas están vendiendo sus casas a bajo precio para obtener compradores lo antes posible y con este dinero migrar. Esto crea una diferencia entre dos tipos de vendedores, los que quieren vender rápidamente su casa y los que quieren vender su casa al precio real, lo cual puede crear inconsistencia en los datos pues casas similares pueden tener precios muy variados. Durante una iteracion previa a la final se entrenó el modelo usando 1/3 de los datos finales, con ello se alcanzó un MAPE de 75 %.El modelo final mejoró un 12 % con esto, demostrando la necesidad de una conjunto de datos variado y grande para obtener un mejor resultado.

9. Conclusiones

Se realizó el entrenamiento de dos modelos de aprendizaje automático para predecir el precio de viviendas de toda Cuba. No se pudo obtener una suficiente cantidad de datos.A pesar de que los datos se ven consistentes se teoriza que son insuficientes. Con el modelo de Keras, antes que hiciera overfiting, se alcanzó un mean aboslute error de 19257 dólares y un mean aboslute percentual error de aproximadamente 63 %.Usando el modelo de XGBoost alcanzamos un mean aboslute error de 32768 dólares y un mean aboslute percentual error de 123 %. Se teoriza que la causa de tal impreciso resultado se debe a la carencia de datos, la extracción de features y/o mala representación de las features, para ello proponemos lo siguiente para mejorar los resultados: - Terminar de hacer scrapping y bajar todos los anuncios. - Buscar features que falten por añadir. - Buscar mejor representación para los datos categóricos. - Buscar mejores hyperparámetros para los modelos.

10. Recomendaciones

Para remediar los problemas enumerados en la sección 8 se propone:

1. Descargar más datos de entrenamiento, para ello se sugiere crear un programa que se ejecute diariamente y que descargue nuevos anuncios publicados en revolico.com. Debido a que este sitio web cambia los nombres de las clases del HTML para proteger sus datos, es necesario que el programa se actualice cada cierto tiempo o que se adapte a dichos cambios.

2. Buscar otros sitios webs de los cuales descargar datos, se recomienda porlalivre.com y detrasdelafachada.com.

3. A la hora de descargar los anuncios usar la fecha de publicación del mismo para en caso de que el precio esté dado en CUP (peso cubano) convertirlo a dólares con el respectivo cambio en el momento de la publicación. En el periíodo de tiempo en que se realiza esta investigación no se cuenta con ningún lugar para identificar el precio en el mercado informal del dólar. El periódico independiente eltoque.com ha guarado durante el último año estos datos y ha informado que proximamente liberará un API para acceder a dichos datos.

4. A pesar de que se eliminaron los datos repetidos (lexicográficamente iguales), algunos usuarios realizan ligeros cambios en sus anuncios. Se recomienda buscar algún método para eliminar estos anuncios también.

5. Muchos usuarios usan precios falsos para llamar la atención e indican que se les contacte para obtener el precio real, se recomienda eliminar dichos anuncios.

6. Los errores ortográficos son extremadamente comunes, por tanto se recomienda buscar una mejor métrica de error al ver si dos palabras indican lo mismo, se recomienda usar comparación fonológica o una distancia de Levenshtein ponderada.

7. Se recomienda buscar otra forma de extraer características y buscar algunas que pudieran faltar y ser importantes para el precio.

8. Se sugiere utilizar una representación de los datos categóricos distinta al one-hot encoding debido a que este crea una matriz esparcida casi vacía.

Referencias

[1] Joyce Manchester. *Inflation and housing demand: A new perspective* https://www.sciencedirect.com.

[2] V.Limsombunchao. *House price prediction: Hedonic price model vs.artificial neural network* https://www.researcharchive.lincoln.ac.nz.

[3] Zhishuo Zhang. *Decision Trees for Objective House Price Prediction* https://ieeexplore.ieee.org.

[4] T.Chen. *XGBoost: A Scalable Tree Boosting System* https://www.kdd.org.

[5] X Xu. *House price forecasting with neural networks* https://www.sciencedirect.com.

[6] Bo Zhao. *Web Scrapping* https://www.researchgate.net.

[7] Anish Chapagain. *Hands-On Web Scraping with Python: Perform advanced scraping operations using various Python libraries and tools such as Selenium, Regex, and others* https://www.books.google.co.uk.

[8] Shengnan Zhang; Yan Hu; Guangrong Bian. *Research on string similarity algorithm based on Levenshtein Distance* https://www.ieeexplore.ieee.org.

[9] Kedar Potdar; Taher S. Pardawala; Chinmay D. Pai. *A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers* https://www.researchgate.net.

[10] Tianqi Chen, Tong He. *xgboost: eXtreme Gradient Boosting* https://www.cran.microsoft.com.

[11] Vladimir Nasteski. *An overview of the supervised machine learning methods* https://www.researchgate.net.

[12] Carl Kingsford , Steven L Salzberg. *What are decision trees?* https://www.nature.com.

[13] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi , Qianli Ma *A survey on ensemble learning* https://www.link.springer.com.

[14] Naomi Altman, Martin Krzywinski *Ensemble methods: bagging and random forests* https://www.go.gale.com.

[15] Chris M. Bishop *Neural networks and their applications* https://www.aip.scitation.org.