International Journal of User-System Interaction 14(4) 2021, 131-147

Artificial Intelligence technologies for enhancing real estate apps

Filip Mănișor[1], Mihai Mașală[1], Eduard Cojocea[1], Traian Rebedea[1,2]

[1] IMOPEDIA SRL, 52 Ion Băiulescu Street, Bucharest, Romania
[2] University Politehnica of Bucharest, Faculty of Automatic Control and Computers, 313 Splaiul Independentei, Bucharest, Romania
E-mail: filip.manisor@gmail.com, mihaimasala@gmail.com, iedi.cojocea@gmail.com, traian.rebedea@upb.ro

Abstract. Given the large amounts of data available in real estate applications and also influenced by the recent uptake of Artificial Intelligence (AI) in various domains, real estate apps are also exploring how AI can be used to improve user experience and provide better functionality for buyers, sellers, and real estate agents. We provide an overview of using various machine learning technologies for several tasks relevant to real estate apps. The experiments were performed using data from IMOPEDIA, one of the leading real estate portals in Romania, and the developed technologies are currently being integrated into the IMOPEDIA app. Thus, we analyze the performance of machine learning models for price prediction and detection of duplicate listings, together with some insights about the explainability of these models. We also show how to visit and click logs can be used to recommend properties similar to the preferences of a user and also to improve search by providing a personalized search.

Keywords: machine learning, real estate apps, price prediction, duplicate detection, page recommendations

DOI: 10.37789/ijusi.2021.14.4.2

1. Introduction

Artificial Intelligence (AI) has seen wide adoption in recent years in many domains, ranging from agriculture to medicine, including various domains such as finance, education, oil and mining, manufacturing, and public administration (Zhang et al., 2022). The main powerhouse of the recent uptake of AI has been the development of more powerful Machine Learning (ML) methods, usually powered by large amounts of data and by the rise of deep neural networks. In this respect, most domains that have access to large

amounts of data have started to incorporate ML and AI methods to improve their products.

To this aspect, it should be no surprise that many real estate and proptech applications (web, mobile, or portal) worldwide have sought to use ML algorithms to improve the features they provide to their users: sellers, buyers, or real estate agents. Among these, Zillow[1], the largest real estate website in the United States by the number of visitors, is probably the most dedicated app to using ML, AI, and other data analytics. It offers price prediction for real estate listings using ML models, but also launched a new feature in 2021, called Zillow Offers, that made buying offers automatic to some sellers. Zillow Offers also employed another ML-predicted estimate for the price that would be accepted by a seller to close a deal.

In this paper, we tackle similar problems but are using Romanian real estate apps from various sources, including IMOPEDIA[2], one of the largest real estate apps in Romania. Thus, we propose solutions using Machine Learning algorithms for three very relevant tasks for real estate apps: price prediction, duplicate listing detection, and page/listing recommendations for a visitor. For all three tasks, our ML models achieve very good results that make them ready to be deployed in production and tested with real users in the following months.

The research presented in this paper is part of the project "IMOPEDIA – Innovative system using Artificial Intelligence for Real Estate Apps" (in Romanian, "IMOPEDIA - Sisteme innovative de Inteligencia Artificial, co-funded by the European Regional Development Fund, Competitiveness Operational Programme, SMIS 129132, POC/524/2/2, action 2.2.1, call nr. 2.

2. Related work

2.1 Price prediction

Price prediction is one of the fundamental tasks in the real estate market. Most often the prediction of the sale price for a property is done by experts, using their past knowledge and comparative market analysis. However, several

[1] Zillow, https://www.zillow.com/, last accessed 15th July 2022
[2] IMOPEDIA.ro, https://www.zillow.com/, last accessed 15th July 2022

automated methods have been proposed for handling this task, if not as a replacement, at least as an assistant for the experts.

An essential step for automated predictions is preprocessing the data. Special care has to be given to missing values, which occur frequently in real-life datasets. Jadhav, Pramod & Ramanathan (2019) discuss the different techniques for single imputation of missing data. Replacing it with the mean value is easiest, however, regression imputation helps maintain the distribution of the data. Feature selection can improve the speed and performance of the prediction and can avoid overfitting. Mutual information is one of the main techniques for this, as discussed by Zhang et al. (2017). Lu et al. (2017) use feature engineering to add new features and perform feature selection using the Lasso algorithm. Landberg (2016) studies the effect different features can have on the price of properties in the Swedish market, directly or indirectly. Usman, Lizam & Adekunle (2020) study the effect of splitting the housing market into more homogeneous submarkets on the price prediction.

Many different Machine Learning approaches have been proposed for price prediction. Abdulal & Aghi (2020) compare regression models Lasso, Ridge, and Random Forest with the performance of Artificial Neural Networks (ANN). Shinde & Gawande (2018) use several scoring methods to compare the performance of Logistic Regression, Support Vector Machines (SVM), Lasso, and Random Forest. They conclude that Random Forest produces the least errors. Ja'afar, Mohamad & Ismail (2021) also point to Random Forest as the best option for prediction, followed by Gradient Boosting and SVM, based on a literature review.

Sarip, Hafez & Daud (2016) propose two fuzzy-based models for the task, of which a fuzzy least-squares model produces good results even on a very small dataset. Zurada, Levitan & Guan (2006) compare several non-conventional algorithms (fuzzy logic, ANN, Memory-based Reasoning) to Multiple Regression Analysis. They conclude that the classic regression approach produces the best results.

2.2 Duplicate detection

Duplicate detection is a fundamental task in the real estate domain as there is a direct negative correlation between the number of duplicate listings on a website and customer satisfaction. One of the most common reasons for user irritation is finding multiple listings (with slightly different details, with

different prices) for the same real-estate property. While real-estate duplicate detection is a heavily understudied field, general methods for finding duplicate or near-duplicate web pages have been widely studied. Henzinger (2006) studies classical algorithms based on string hashing for detecting near-duplicate web pages. Similarly, Theobald et al. (2008) introduce another purely textual matching algorithm for detecting near-duplicate web pages. Yandrapally et al. (2020) perform an extensive study of multiple near-duplicate detection algorithms for web crawling. Their findings suggest that no near-duplicate algorithm is particularly suited for the task at hand and that application-specific knowledge greatly improves the performance of the considered algorithms. In this work, we devise a specialized duplicate detection algorithm for the real estate domain that integrates expert-level real estate knowledge.

2.3 Page recommendations

The task regarding page recommendations is referring to models which are able to recommend pages that are a good match for a user and to be able to sort the results of a query search such that the most relevant pages for a user are listed at the top.

Recommender systems are a vital part of any web platform, regardless of the platform's profile and focus. Being able to serve the most relevant content for a specific user is achieved via many possible solutions.

Covington et al. (2016) propose a solution for YouTube consisting of a neural network for candidate generation and one for ranking. The first network is able to extract from a large number of videos a few hundred videos which are considered generally of interest for a user. The second network is used to present to the user to top few best videos based on the results of the first network.

Fu et al. (2015) present a solution for ranking real estate from a mixed land use viewpoint, where geographical, community, neighborhood features, and other urban functions are taken into consideration.

Gharahighehi et al. (2021) surveyed available solutions for implementing recommender systems for real estate, underlining the inherent difficulty of doing so due to the many domain-specific limitations in comparison to other web platforms such as video platforms, newspapers, or social media.

3. Proposed methods

3.1 Price prediction

The price prediction was done using Machine Learning algorithms, which depend in large part on the quality of the input data. Thus, the preprocessing and feature operation steps were done before the actual prediction.

The initial dataset used for price prediction was composed of a bit over 100,000 listings of properties with verified sale prices by experts, obtained from the MLS (Multiple Listings Service) - FlexMLS[3] and it contained data regarding properties in Bucharest and Constanta. After the preprocessing step, a subset of 11967 data points was extracted from the initial set. These final listings were all associated with properties that were sold between 2015 and 2020 and contained information regarding both the properties and the sale.

3.1.1 Preprocessing

Missing data

Much of the input data was incomplete, either because some features did not apply to certain instances or because they were not mandatory. Several of the more common methods for handling missing data were tested, from using the mean value to using the k-Nearest Neighbors (k-NN) algorithm to estimate missing values using the values of the closest k neighbors of the instance. A custom method was also designed, which used domain knowledge to deal with missing data in most of the features, setting their default values. In the cases where a default value did not make sense, the k-NN algorithm was used.

Data transformation

All features were transformed into a numerical format before running the prediction algorithms. Boolean labels were transformed into binary data, while categorical features were transformed using One Hot Encoding.

For features where the number of possible labels was considered too large, a particular case of One Hot Encoding was used. The most frequent labels were transformed into new binary features, while the rest of the labels were

[3] FlexMLS in Romania, https://www.mls.ro/, last accessed on 15th July 2022

all grouped into separate binary features.

For date features, each date was assigned a number, representing the interval it is part of, starting from the oldest date in the dataset. Three-month intervals were chosen to better represent the seasonality of the data. Four new hot encoding features which represent the four quarters of the year were also added to model the potential seasonality of the data.

Outliers

Most outliers were due to human-entry mistakes in the input data and they degraded the performance of the prediction algorithms. For each feature, all the possible values in the input data were considered and outlier values were detected using the Inter Quartile Range method and a large factor, so as to only flag obviously incorrect values. The instances which had outliers were removed from the dataset, leading to 0.5% of total instances being removed.

3.1.2 Feature operations

Feature selection and adding new features were done in order to obtain more powerful predictive models.

New features

The input data were augmented with new features extracted from other sources, in order to add information for the prediction algorithm.

One set of new features was formed by the distance between each property and the main points of interest in the city. The points of interest included airports, rail stations, and the cardinal points of the city.

Then properties were assigned to cells on a map and OpenStreetMap (OSM) was used to calculate new features per cell. Points of interest provided by OSM were grouped into several relevant categories (such as food, culture, and lower or higher education) and their counts were used as new features. Two other new features were constructed using OSM: the house percentage and the average floor for each area. A new *cluster* feature was also added by grouping similar cells into clusters.

Feature selection

Some of the features were not relevant to the price prediction or were heavily correlated with other features, so they did not provide any benefit.

The most important features were selected using the Recursive Feature Elimination (RFE) algorithm, which produces the N most important features.

For the estimator model, a Decision Tree regression algorithm was used.

3.1.3 Prediction

A series of algorithms were tested for the prediction task. Elastic Net (EN) was used to obtain a baseline for the results of the more complex models. The rest of the models were chosen based on the state-of-the-art literature review and domain knowledge.

The chosen algorithms were Support Vector Regressor, K-Nearest Neighbors, AdaBoost, Gradient Boosting, and Random Forest. The prediction results for these algorithms can be seen in Section 4.1.

3.2 Duplicate detection

The duplicate detection module's role is to automatically decide if two entries represent the same property. We resort to Machine Learning algorithms to solve the task at hand, with the final version of the module based on Random Forests. Therefore, handling and preprocessing the data into the appropriate format is essential. This process is deeply intertwined with the data analysis step as not all the data features are relevant to the task at hand. As we had no specific data for duplicate detection, the next step was building a simple graphical interface for annotating duplicate entries. Having built a corpus of duplicate entries, we trained ML models for detecting duplicate entities.

3.2.1 Data preprocessing and analysis

We start with a corpus of real estate listings from the IMOPEDIA app, containing a total of 54.564 listings with 118 different features for each entry.

The first step was selecting the relevant features for the duplicate detection task out of the 118 features present in the corpus. Using the expertise of professional real estate agents and data analysis techniques we managed to narrow the relevant features down to 18. These features include the year in which the building was built, the text of the listing, the number of bathrooms, the number of rooms, the location, the asking price or the type of property (a house or an apartment), or the total area.

3.2.2 Data annotation

After removing perfect matches (based on the 18 features) we were left with a corpus of 54.138 real-estate listings. We want our models to be able to

distinguish between two properties that are similar (i.e., in a similar location, with a similar total area) but still different. For example, we don't want to annotate a pair of listings that represent properties in different cities, or a one-bedroom apartment with an apartment with 4 bedrooms, as they clearly represent different properties.

To alleviate this issue, we build an intermediate corpus that contains possible duplicates. We aim to select pairs of listings that are similar, listings that are considered candidates to represent the same property. To build this corpus we devise a set of rules based on each feature that is successively applied to decide if two listings are possibly duplicates. Some of the rules are meant to check the county in which the property is situated, the number of rooms, or the asking price. While for some features we require an exact match such as the number of rooms in the county, for other features we look for a more relaxed condition: for the precise location of the property, we require that there is a difference of less than 1000m, for the price we settle for a difference of less than 10%, while for the year in which the building was built a difference of under 2 years is still considered a match. The process of developing the rules and their parameters was intertwined with a data analysis process and input from real estate experts.

The next step was to run the rules on all pairs of listings. To ease the computational resources needed to build the corpus we resort to a process of binning. We apply successive binning based on the following features: county, type of property, number of rooms, number of baths, and floor. Finally, we apply the previously mentioned rules to each bin.

Out of the total 20.473 pairs of possible duplicates, around 80% contain differences in the text of the listings, 73% contain differences in the asking price, 55% contain differences in the surface and only 30% contain differences in the building year.

We developed a simple graphical interface to ease this process. Our goal was to make this process as easy as possible for the human annotator and as such our interface is made up of three main visual components: a central interface built for handling input from the annotator and large parallel components in which we display the page of each listing in a pair. The human annotator, when faced with a pair of listings has three labeling options: both listings refer to the same property so they (the listings) are duplicates, or the listings refer to different properties in which case they should be labeled as different. Finally, if not enough information is available to make an informed

decision, we provide an additional label in the form of "inconclusive".

In the first step, we annotate a total of 600 pairs. As we analyzed the data, we found that based on the number of differences we need to focus on categories one, two, three, and four differences. The other categories contain straightforward cases: for 0 differences we can safely label all pairs as duplicates and for five or more differences we can safely say they are not duplicates. In all further experiments, we will use data from categories one, two, three, and four. Finally, this leaves us with a corpus of 300 annotated pairs with 157 duplicates, 52 inconclusive, and 91 different pairs.

3.2.3 Detecting duplicates

After we built the corpus, we turn our attention to training and evaluating models for duplicate detection. In all our experiments we will use a 10-fold cross-validation with 5 splits, therefore generating for each fold of 240 train samples and 60 test samples. For most of the features, we will use the absolute difference between the two values of that specific feature. For example, the difference between the asking price of two listings becomes one of the features of our model. For other features, we apply more complex rules to transform the data into the appropriate format. The corpus contains the location of a property in the form of latitude and longitude, which is transformed into a feature in the form of the distance in meters between two real estate properties. Another interesting example is represented by the text column. As the text in the listings varies from semi-structured text to whole stories, we used lexical models and semantic models to compute the similarity between the two texts. We experimented with simpler semantic models such as Word2Vec embeddings (Mikolov et al., 2013) and with more complex language models in the form of Transformers (Vaswani et al., 2017, Devlin et al., 2018), namely RoBERT models (Masala et al., 2020).

3.3 Page recommendations

The recommender system we propose contains multiple methods: a learning-to-rank model which sorts a list of pages according to a score, a Siamese model with triplet loss which outputs how similar two pages are and a collaborative filtering model which outputs how appropriate is a page for a certain user.

3.3.1 Data preprocessing and analysis

For training the models we used two datasets. The first dataset consists of more of the corpus of real estate listings described in the previous subsection. The second dataset consists of more than 1 million unique site visits, both from real users and bots, containing the IP address of the user, the date and time of the visit, the URL, and the id of the listing contained by the page.

The first dataset for preprocessed for outliers and missing values using the methods mentioned in Section 3.1. Preprocessing the second dataset consisted of identifying and removing the bots from the visits since they do not add any value to the data. Thus, all the visits have been grouped by IP address and all the visits which are within 1 second of each other for a certain IP address have been attributed to bots. The remaining visits have been grouped into visiting sessions, where multiple visits from the same IP address have been associated with a unique user.

The dataset containing listings serves as the data for the Machine Learning models to interpret objectives and features of apartments and houses, while the dataset containing user visits serves as the data for the Machine Learning models to understand user latent or hidden preferences (which are not necessarily visible in the first dataset) and user interaction with the listings.

3.3.2 Learning to rank

For the user to have a positive experience with a real estate app, the results of each search should be ordered based on how relevant the properties in relation to the search query and the user are, and not only based on some heuristic. Thus, we have trained multiple models which for a property given as input, output a score that tells how relevant that property is.

The dataset used for training consisted of a mixture of the two datasets presented above. The data used as input contained 28 numerical features of a listing and the label was a binary label, which represented if a user is interested or not in the specific ad. We considered that a user is interested in a listing if she spent at least 100 seconds viewing the page of the ad. We estimated the time spent on a page by subtracting the time between 2 visits with the same IP address and removed the differences where the time between visits was anomalous long. Using this data, we have implemented logistic regression models and multiple neural networks.

Also, instead of a binary value for the interest of the user, we placed how interested a user is in 6 classes from 0 to 5, where 0 is the least interested and 5 is the most interested. Thus, the labels of the data change and we consider

the class based on the time spent with the following thresholds: 10 seconds, 30 seconds, 60 seconds, 120 seconds, and 300 seconds.

3.3.3 Triplet loss

In order to offer the user the best results for a search query, the recommender system should be able to determine how similar 2 properties are based on the user's page visits.

We have implemented a Siamese network with 3 hidden layers and a dropout layer, using a triplet loss for evaluating the model. The input of the network consisted of a triplet of listings, one of which is the anchor, another one is a positive example and the third is a negative example. The anchor represents a listing to which we compare the other 2 listings. The positive example represents a listing that is considered to be of a similar value for the user as the anchor. The negative example represents a listing that is considered to be of a different value than the anchor. Thus, we created another dataset that contained these triplets. We started from the user visits sessions and considered one listing as the anchor, and another listing from the visit session as the positive example (since the user visited both pages in the same visit session he is likely taking a similar interest in both pages) and for the negative example we considered a random listing which is not in the current visiting session. Each listing in the triplet consists of 14 relevant numerical features.

3.3.4 Collaborative filtering

We implemented a collaborative filtering model to be able to tell how relevant the results of a search query are for a user based on his past visits and other users' past visits, without taking into consideration the features of an ad. In order to achieve this, we take into consideration only those users who have at least 20 visits. Thus, the model could have a good grasp of the user's profile. If a user has fewer visits, his search results will be heuristically ordered until he makes enough visits.

The dataset used will contain a list of triplets: user, listing, and rating, where user and listing are unique ids and rating is a value from 1 to 5. The value of the rating is computed similarly as in section 3.3.2, where the value 1 represents a low interest and the value 5 represents a high interest of the user. The value is based on the time spent by the user on the page, having the value computed based on the following time thresholds: 10 seconds, 30

seconds, 120 seconds, and 300 seconds.

4. Results

4.1 Price prediction

Testing was done to determine both the best set of features and the actual prediction algorithm. Only the most relevant results will be selected here, due to space constraints.

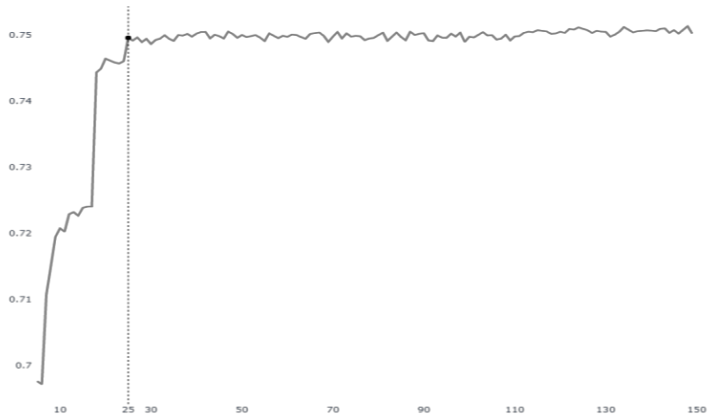

Figure 1 – R^2 scores using a different number of features for price prediction

The feature selection was done using the mutual information score for the features, averaging over 10 runs for more accurate results. After all features were ranked according to their feature importance score, the optimum number of features was selected based on the result of predictions using the Elastic Net algorithm, which provides a good baseline and is also quite sensitive to feature selection.

Figure 1 shows the coefficient of determination (R^2) scores for each number of features, obtained by running 5-fold cross-validation 5 times and averaging the results. The final number of selected features was 25, as only minimal improvements were obtained by adding any more features.

The results for the prediction algorithms were obtained by averaging 10 runs of 5-fold cross-validation, in order to minimize the variations that can

occur between different training sessions. Table 1 contains the scores for Percentage Mean Average Error (pMAE), Mean Average Percentage Error (MAPE), Root Mean Squared Error (RMSE), and Root Mean Squared Logarithmic Error (RMSLE). The best results were obtained using the Random Forest algorithm.

Table 1 – Results of price prediction algorithms

Algorithm	pMAE	MAPE	RMSE	RMSLE
SupportVector	0.3314	0.3700	30649	0.4413
kNN	0.1405	0.1411	14376	0.1862
AdaBoost	0.1892	0.2593	14618	0.2901
GradientBoost	0.0863	0.0880	9367	0.1194
HistGradientBoost	0.0645	0.0649	7208	0.0924
RandomForest	0.0520	0.0506	6357	0.0806

4.2 Duplicate detection

In this section, we present the results of the duplicate detection module. As previously mentioned, we experiment with Random Forest models, using 10-fold cross-validation and averaging the results over 10 runs. To recap the model gets as input a vector of differences or similarities (computed on a pair of listings) and must assign to each entry one of the labels: duplicate, inconclusive or different. In Table 2 we present the core of our results. In the first column, we mark the text similarity function used (in the first row where the text similarity function is “None” we ignore the text entirely. In the following columns, we present the common Precision, Recall, and F1 metrics on the “duplicate” class and a global metric in the form of accuracy.

Table 2 – Results of duplicate detection methods

Text similarity	Precision	Recall	F1	Accuracy
None	0.76	0.81	0.78	0.63
Word2Vec	**0.82**	0.86	**0.84**	0.69
RoBERT-small	0.80	0.84	0.82	0.67
RoBERT-base	0.80	**0.87**	0.83	0.68
RoBERT-large	0.79	0.85	0.82	0.66
All	0.81	0.81	0.81	**0.70**

We used all text similarity functions as a mean of feature extraction, neither function is further trained on the current corpus. We note that while there are no significant differences between the used text similarity function, using text

information clearly boosts performance across all metrics. As no model has the best performance across the used metrics, we pick the model based on Word2Vec as the "best" in part due to its simplicity (especially when compared to BERT-based models). Also, it is the best-performing model on the F1 metric for the "duplicate" class, the class we are most interested in.

In Figure 2 we present the importance of each feature for predicting if two listings are duplicates or not. We note the most important in their respective order: text information, surface, asking price, and location.

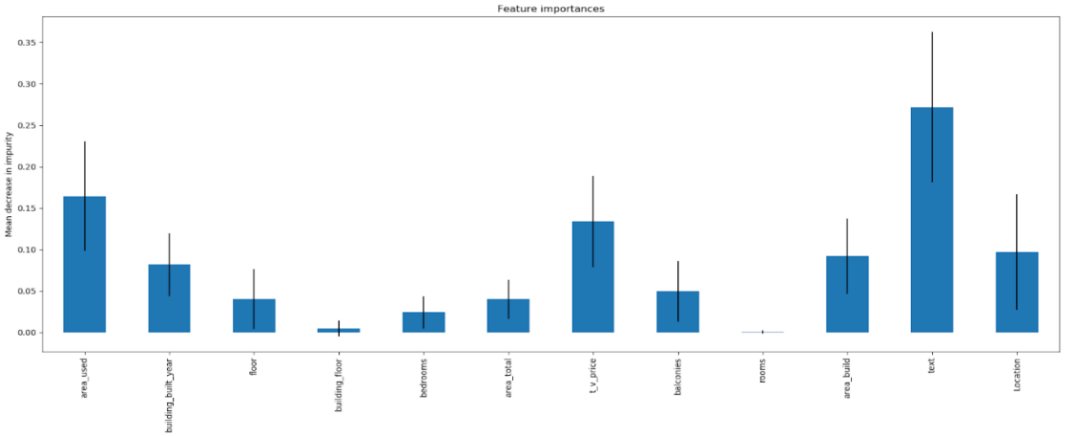

Figure 2 – Feature importance for duplicate detection

4.3 Page recommendations

In this section, we discuss the results of the page recommendation. In Table 3 we present the results for the models we have implemented for learning importance scores for listings. We have tried several models, but neural networks with more than 2 hidden layers yielded weak results, hence we included only those in the table.

Table 3 – Results of page recommendations models

Model	Input features	Accuracy	Precision	Recall	F1
2 hidden layers NN	28	0.67	0.59	0.79	0.68
2 hidden layers NN	16	0.73	0.70	0.82	0.75

2 hidden layers NN + dropout	28	0.83	0.83	0.86	0.84
2 hidden layers NN + dropout	16	0.78	0.77	0.82	0.79
Logistic regression	28	0.76	0.71	0.84	0.77
Logistic regression	16	0.74	0.7	0.79	0.74

For the Siamese model, the dataset contained more than 800k triplets (anchor, positive, negative), which were split into 70% for training, 20% for validation, and 10% for testing. The results are harder to interpret than normal since the ground truth is subject to user preferences and thus there is some overlap between what could be a positive example and a negative example. We evaluate how many of the triplets have a bigger cosine similarity between the resulting embeddings of positive and anchor than the cosine similarity between the resulting embeddings of negative and anchor. Thus, out of the 88 376 triplets used for testing, 65 601 had more similar embeddings for anchor and positive than anchor and negative.

5. Conclusions

In this paper, we have discussed several methods for incorporating Machine Learning methods into real estate apps in order to solve three relevant tasks: price prediction, detection of duplicate listings, and page/listing recommendations. The proposed methods achieve good results, showing that ML methods have the performance required to be integrated into real estate products for the Romanian market. In the forthcoming months, the developed ML models will be integrated into the user-facing IMOPEDIA app.

While these tasks seem very technical, they are very related to the user's needs: buyers do not want fake or duplicate listings and are interested in relevant price predictions and relevant other listing recommendations related to their needs and interests. Also, the results should also be useful for Computer-Human Interaction practitioners that want to assess the performance of Romanian real-estate apps, maybe also focusing on the technology usage component and the level of integration of ML algorithms into these apps.

Acknowledgments

The research presented in this paper is part of the project "IMOPEDIA – Innovative system using Artificial Intelligence for Real Estate Apps" (in Romanian, "IMOPEDIA - Sisteme inovative de Inteligenta Artificiala in domeniul portalurilor imobiliare"), co-funded by the European Regional Development Fund, Competitiveness Operational Programme, SMIS 129132, POC/524/2/2, action 2.2.1, call nr. 2.

References

Aghi, N., & Abdulal, A. (2020). *House Price Prediction*. Technical Report. Khristianstad University Sweden.

Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 191-198). ACM.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fu, Y., Liu, G., Papadimitriou, S., Xiong, H., Ge, Y., Zhu, H., & Zhu, C. (2015). Real estate ranking via mixed land-use latent models. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 299-308). ACM.

Gharahighehi, A., Pliakos, K., & Vens, C. (2021). Recommender Systems in the Real Estate Market—A Survey. Applied Sciences, 11(16), 7502.

Henzinger, M. (2006, August). Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 284-291).

Ja'afar, N. S., Mohamad, J., & Ismail, S. (2021). Machine learning for property price prediction and price valuation: a systematic literature review. *Planning Malaysia*, 19.

Jadhav, A.S., Pramod, D., & Ramanathan, K. (2019). Comparison of Performance of Data Imputation Methods for Numeric Dataset. *Applied Artificial Intelligence*, 33, 913 – 933.

Landberg, N. (2016). The Swedish Housing Market: An empirical analysis of the real price development on the Swedish housing market.

Lu, S., Li, Z., Qin, Z., Yang, X., & Goh, R. (2017). A hybrid regression technique for house prices prediction. *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 319-323.

Masala, M., Ruseti, S., & Dascalu, M. (2020, December). Robert–a Romanian bert model. In *Proceedings of the 28th International Conference on Computational Linguistics* (pp. 6626-6637).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Sarip, A. G., Hafez, M. B., & Daud, M. N. (2016). Application of fuzzy regression model for real estate price prediction. *Malaysian Journal of Computer Science*, 29(1), 15-27.

Shinde, N., & Gawande, K. (2018, October). Survey on predicting property price. In 2018 *International conference on automation and computational engineering (ICACE)* (pp. 1-7). IEEE.

Theobald, M., Siddharth, J., & Paepcke, A. (2008, July). Spotsigs: robust and efficient near duplicate detection in large web collections. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 563-570).

Usman, H., Lizam, M., & Adekunle, M. U. (2020). Property price modeling, market segmentation, and submarket classifications: A review. *Real Estate Management and Valuation*, 28(3), 24-35.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Yandrapally, R., Stocco, A., & Mesbah, A. (2020, June). Near-duplicate detection in web app model inference. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering* (pp. 186-197).

Zhang, Y., Yang, A., Xiong, C., Wang, T., & Zhang, Z. (2014). Feature selection using data envelopment analysis. *Knowledge-based systems*, 64, 70-80.

Zhang, D., Maslej, N., Brynjolfsson, E., et al. (2022). *The AI Index 2022 Annual Report*. AI Index Steering Committee, Stanford Institute for Human-Centered AI, Stanford University.

Zurada, J. M., Levitan, A. S., & Guan, J. (2006). Non-Conventional Approaches To Property Value Assessment. *Journal of Applied Business Research (JABR)*, 22(3).