Utilizing the UK Property Price Dataset for Market Analysis using PySpark and DataBricks

Qiao Qin
MPML
Email: qqin@umd.edu

***Abstract*—Leveraging the distributed computing capabilities of PySpark within the Databricks platform, I analyzed the UK Property Price Dataset, consisting of 1.5 million rows, for market insights. My study aims to uncover the valuable insights within the real estate market. The research methodology involved the utilization of PySpark's parallel and distributed computing capabilities to process big data efficiently. I addressed the price trends over years, the impact of geographic location on housing price, and other valuable aspects. Multiple machine learning models were used in this experiment to predict house prices. The results contribute to a better understanding of market trends in the real estate sector.**

Keywords: UK Property Price Dataset, PySpark, Databricks

I. Introduction

The real estate market is a dynamic ecosystem influenced by various factors, including economic conditions, geographical location, and temporal trends. This paper embarks on a comprehensive analysis of the UK Property Price Dataset, exploring its diverse features to derive meaningful insights and predictions regarding housing prices.

One aspect of my approach is the utilization of PySpark and Databricks, empowering me to process big data. Within this framework, I employed machine learning algorithms such as Gradient Boost Decision Tree, Random Forest and Linear Regression, among others, to conduct a comparative analysis of their performance. This evaluation uses metrics such as predictive accuracy, specifically Mean Square Error and R-Squared, and computational time, providing a comprehensive understanding of the strengths and weaknesses of each algorithm.

In subsequent sections, I delve into the dataset's intricacies, outline my methodology, and present the results derived from the analyses. These results lead to a robust exploration of the UK property market, providing actionable insights for stakeholders.

II. Intro to the Dataset

The dataset I will be using is *UK Property Price official data 1995-202304*. This dataset provides comprehensive information on property sales in England and Wales, as sourced from the UK government's HM Land Registry. It offers valuable insights into property transactions, including sale prices, locations, and types of properties sold. This dataset is particularly useful for analysts, researchers, and businesses looking to understand market trends, property valuations, and investment opportunities in the real estate sector of England and Wales.

This is a 4.82G structured dataset, with 28 million rows and 16 columns. The meaning for each column is:

- Transaction unique identifier
- Price: The price for which the property was sold.
- Date of Transfer
- Postcode: The postal code where the property is located.
- Property Type
- Old/New
- Duration
- PAON (Primary Addressable Object Name): Typically the house number or name.
- SAON (Secondary Addressable Object Name): Additional information if the building is divided into flats or sub-buildings.
- Street: The street name where the property is located.
- Locality: Additional locality information.
- Town/City: The town or city where the property is located.
- District: The district in which the property resides.
- County: The county where the property is located.
- PPDCategory Type
- Record Status - monthly file only

The dataset contains records of property sales dating back to January 1995, up to April 2023. Due to computational resource limitations, I will randomly select 500,000 samples for each year from 1995 to 2022 and include all data from the year 2023, forming my dataset with a total of 1,528,903 samples.

III. Exploratory Data Analysis

A. Housing Price

In this section, I conduct a comprehensive Exploratory Data Analysis (EDA) to gain insights into the UK Property Price Dataset. The analysis involves examining various features, illustrating their distributions or trends, and providing insightful interpretations.

To understand the overall trend of housing prices, I present a time-series analysis of the dataset. The following plots show the average prices from 1995 to 2023, the average prices from January to December, and the average prices changes in one month.

Fig.1 reveals a consistent upward trend in average house prices from 1995 to 2023. Initially, I intended to use the

Fig. 1

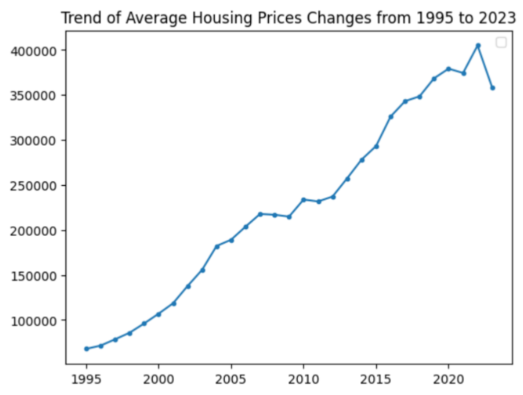


Fig. 2

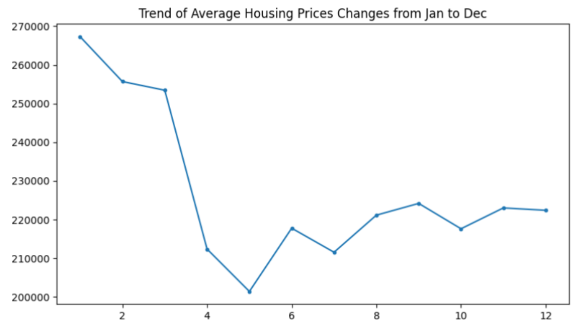
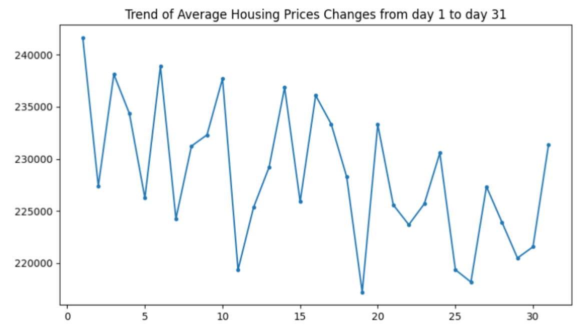


Fig. 3

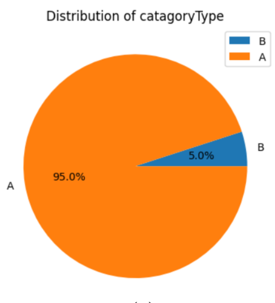


portion of the dataset from 1995 to 2022 as the training set and the 2023 portion as the test set. However, there is a noticeable decrease between 2022 and 2023, making this partitioning seem impractical. Therefore, I chose a random split of the dataset.

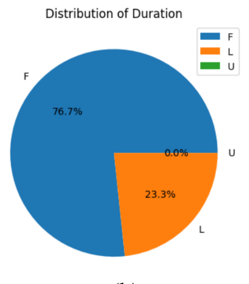
From Fig.2 and Fig.3, it is evident that the distribution of house prices across months and dates is not uniform, showing noticeable fluctuations. Therefore, both month and date make as non-negligible features influencing house prices.

B. Categorical Features

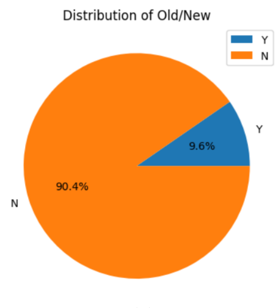
Fig. 4 are the distribution of values for some categorical variables. From these pie charts, it can be observed that there is some imbalance in the distribution of almost every feature here. However, since the imbalance is in the distribution of



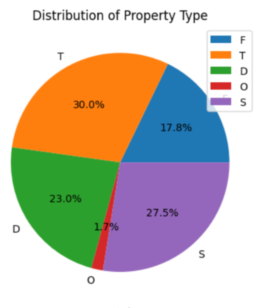
(a)



(b)



(c)



(d)

Fig. 4: Feature distribution

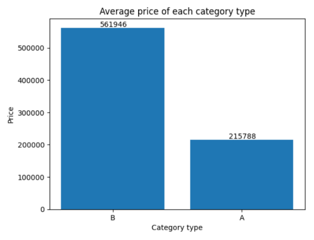
features rather than the price I want to predict, it is not necessary to handle it.

Fig. 5 are the average housing price of each feature. From these bar charts, it is evident that for these categorical features, the distribution of housing prices varies across different feature values. Specifically, for the features old/new and duration, the price differences between different values are relatively small, suggesting that these two features may contribute less to the prediction. On the other hand, for categoryType and propertyType, the price differences between different values are substantial, indicating that these two features likely play a more significant role in influencing predictions.

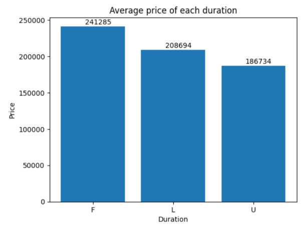
C. Location

The features representing geographical locations in the dataset include the following: postcode, street, locality, town/city, district and country. Among these features, some are categorical variables with a large range of values, making it hard to extract meaningful visualizations. Therefore, only country is used as a representative feature. The bar plots below show the top 10 regions with the highest house prices and the highest number of houses sold. From the graph, it can be observed that there are differences in house prices across different regions, with Great London having the highest house prices and the highest number of houses transactions.

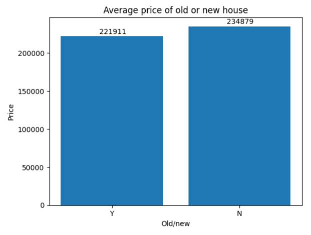
Top 3 high-priced country: GREATER LONDON, WEST NORTHAMPTONSHIRE, WINDSOR AND MAIDENHEAD. Top 3 frequent country: GREATER LONDON, GREATER MANCHESTER, WEST MIDLANDS.



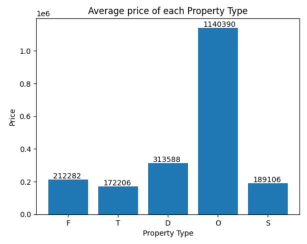
(a)



(b)



(c)



(d)

Fig. 5: Average housing price of each feature

Fig. 6

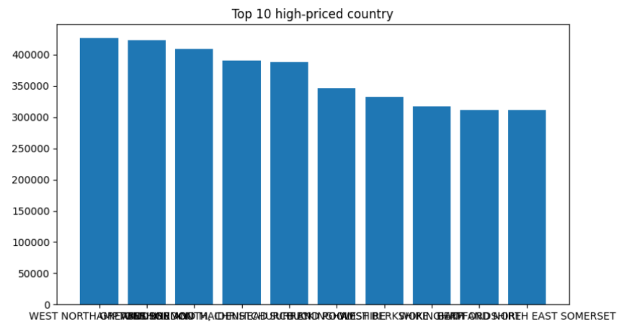
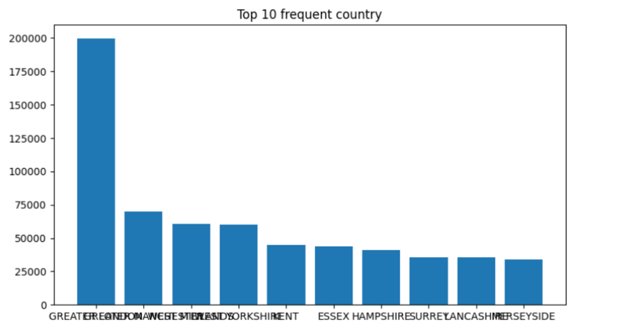


Fig. 7



IV. Data Preprocessing

Data preprocessing is a critical phase in preparing the raw dataset for analysis and modeling. In this section, I outline the steps taken to enhance the quality, consistency, and interpretability of the data.

A. Handling Missing Values

The table above illustrates the count of non-null values and the value count for each feature. It is evident that the majority of features do not have missing values. However, two features exhibit a significant number of missing values. Among the total 1,528,903 samples, SAON has only 180,715 non-null values, and locality has 913,570 non-null values. Consequently, these two features were discarded due to the high prevalence of missing values.

Dataset info		
Feature	Non-null values	Value count
transactionID	1528903	1528903
price	1528903	-
date	1528903	9372
postcode	1526680	704312
propertyType	1528903	5
old/new	1528903	2
duration	1528903	3
PAON	1528619	102298
SAON	180715	9061
street	1504323	214214
locality	913570	18701
town/city	1528903	1167
district	1528903	464
country	1528903	132
categoryType	1528903	2
recordStatus	1528903	1

In addition, for the postcode feature, it is dropped due to the presence of a large number of unique values and geographic information can already be represented by other features such as street and district. For the street feature, as it is a categorical variable, missing values were imputed using the mode.

B. Encode

For the date feature, three new integer features were created to represent the year, month, and day.

As for the propertyType and duration features, with only 5 and 3 unique values respectively, one-hot encoding was applied.

For the old/new and categoryType features, which have only two possible values, direct 0/1 encoding was applied.

For other features like PAON, street, town/city and so on, there are many unique values. The majority of the resulting one-hot encoded matrix will be filled with zeros, as each row corresponds to a single unique value. This creates a sparse matrix, which can be memory-intensive and inefficient for storage and computation, and can increase the risk of overfitting. Therefore, one-hot encoding is not a good choice for these features. Under such circumstances, I apply different encoding.

- Label-encoding: assign each unique value a unique integer label. This reduces dimensionality but may introduce ordinal relationships.
- Frequency-encoding: encode each category with the frequency of occurrences in the dataset. This can capture information about the distribution of categories.
- Mean Encoding: encode each category with the mean of the target variable for that category. This can be useful

when the relationship between the categorical feature and the target variable is informative.

Considering that none of the three encoding methods mentioned above can fully represent the features, in order to preserve as much information as possible, all three encoding methods are applied to these features.

C. Normalization

Normalization is a crucial step in the data preprocessing pipeline, aimed at standardizing the scale of numerical features within a dataset. This process ensures that variables with different units and magnitudes contribute equally to the model training process, preventing larger-scale features from overshadowing smaller-scale ones. Many machine learning algorithms, especially those based on gradient descent, converge faster when dealing with normalized data.

There are two mainstream normalization methods: Standard Scaling and Min-Max Scaling. The formula for Standard scaling is given by:

$$X_{standardized} = \frac{X - X.mean()}{X.std()}$$

And The formula for Min-max scaling is given by:

$$X_{min_max} = \frac{X - min(X)}{max(X) - min(X)}$$

Standard Scaling is employed in this project.

D. PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique widely used in data preprocessing. It aims to transform a dataset into a new coordinate system, where the greatest variance is aligned with the first axis (principal component), the second greatest variance with the second axis, and so forth. This allows for retaining the most critical information in a dataset while reducing its dimensionality.

After encoding, there are 22 features in the dataset, and PCA is particularly useful when dealing with datasets with a large number of features, it can lead to significant computational savings during subsequent modeling.

In this project, when applying PCA, reducing the data's dimensionality to 17 preserves approximately 91.8% of the explained variance, and the computational time for PCA itself is only around 10 seconds. Therefore, PCA is performed with a parameter of k=17.

Note that both normalization and PCA can impact the interpretability of features. While normalization doesn't inherently change the meaning or nature of individual features, it might change the interpretation of coefficients or weights in certain models. PCA transforms the original features into a set of linearly uncorrelated variables. These principal components are linear combinations of the original features. While they capture most of the variance in the data, the resulting components may not have a clear and direct interpretation in terms of the original features.

V. Building Models

In this project, 3 machine learning models are employed to predict housing prices: Gradient Boosting, Random Forest, and Linear Regression. Each model was implemented using PySpark, taking advantage of its distributed computing capabilities. In this section,

A. Models

This subsection briefly introduces gradient boosting regression, random forest regression, linear regression, their advantages and disadvantages compared to each other, and the main hyperparameters that can be tuned in pyspark.

Gradient boosting is an ensemble learning technique where multiple weak learners (typically decision trees) are trained sequentially, with each tree correcting the errors of the previous one. The final prediction is a weighted sum of the predictions from individual trees. Advantages of this algorithm are: high predictive accuracy, the ability to capture complex relationships in the data, and its robust to outliers. Disadvantages are: its tendency to overfitting and higher training time. The hyperparameters are maxDepth (maximum depth of the tree), maxBins (maximum number of bins used for splitting features) and maxIter (maximum number of boosting iterations, or say trees).

Random forest is also an ensemble method that builds multiple decision trees and merges their predictions to obtain a more accurate and stable prediction. Each tree is trained on a random subset of features and data points. Advantages of random forest are: robust to overfitting due to averaging of multiple trees and it provides feature importance scores. While the disadvantages are: it may not perform as well as Gradient Boosting on certain datasets. The hyperparameters are: maxDepth (maximum depth of the tree), numTrees (number of trees in the forest) and maxBins (maximum number of bins used for splitting features).

Linear Regression is a linear approach to modeling the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship and estimates the coefficients that minimize the sum of squared differences between observed and predicted values. Its advantages are: it's simple and interpretable, fast to train and make prediction, and it performs well when the relationship between features and target is approximately linear. And the disadvantages are: it assumes a linear relationship, which may not hold in all cases, and it's sensitive to outliers. The hyperparameters are: regParam (regularization parameter that controls overfitting), elasticNetParam (elastic net mixing parameter that combines L1 and L2 regularization), and maxIter (maximum number of iterations).

B. Methods and Techniques

In the model training phase, two key techniques are employed to enhance model performance: cross-validation and grid search. Cross-validation is a robust validation technique that partitions the dataset into multiple subsets, allowing the model to be trained and validated on different combinations of the data. It helps in the model's generalization performance and reducing overfitting.

Grid search, on the other hand, is utilized to systematically explore a range of hyperparameter values for each machine learning model. By evaluating the model's performance across various hyperparameter combinations, I aim to identify the optimal set of parameters that maximizes predictive accuracy. This exhaustive search method contributes to refining the model's configuration and ensuring its optimal performance.

Additionally, when training the linear regression model, the Polynomial Expansion technique is employed. This involves transforming the input features by adding polynomial terms, enabling the model to capture non-linear relationships between variables. In particular, the linear regression model is extended to higher-degree polynomial features, enhancing its flexibility and capacity to capture complex patterns within the data.

The combination of cross-validation, grid search, and Polynomial Expansion contributes to a thorough and effective model training strategy. However all three methods will significantly increase the training time.

VI. Experimental Result

In the experimental setup, the analysis is conducted within the Databricks environment, utilizing the Databricks Runtime version 14.2. This runtime includes Apache Spark version 3.5.0 and Scala version 2.12, providing a robust and efficient platform for distributed data processing and machine learning tasks. The computational resources is configured with an i3.4xlarge node, featuring 122GB of memory, 16 cores, 4 DBU/h. This environment choice was driven by the need for substantial memory and processing capabilities to handle the large-scale dataset and complex computations inherent in big data analytics.

This table below presents the feature importance calculated by a decision tree with maxDepth=20. This table provides a rough observation of the contributions made by different features to predicting prices. It can be observed that the year feature has the highest importance, confirming the earlier observations in section 3 regarding the increasing trend in prices over the years. Therefore, year is identified as a significant influencing factor. Additionally, the old/new and duration features have relatively low importance, aligning with the observations on these features: the average price differences between different values of these features are small, indicating that they are not significant influencing factors.

As for geographical location, the sum of the values of index, target, and frequency for each feature can be roughly calculated to obtain the contribution of each feature to predicting prices. It can be observed that the four features representing geographical locations, namely street, town/city, district and country all have a significant degree of contribution to price prediction.

Feature importance	
Feature	Importance
year	0.2415
month	0.0877
day	0.1147
propertyType	0.0084
old/new	0.0016
duration	0.0056
PAON index	0.0109
PAON target	0.0007
PAON frequency	0.0108
street index	0.0034
street target	0.1
street frequency	0.0319
town/city index	0.0067
town/city target	0.0599
town/city frequency	0.0397
district index	0.031
district target	0.0145
district frequency	0.021
country index	0.0051
country target	0.0415
country frequency	0.0514
categoryType	0.0188

Two surprising points stand out among these features: Firstly, the importance of street is higher than expected. This may indicate that while overall housing prices may not vary significantly among different towns, cities, districts or countries, there are noticeable differences within different street in the same town. Secondly, the importance of day is unexpectedly high, even surpassing month. This counter-intuitive conclusion could serve as a direction for further exploration in subsequent analyses.

Table 1 presents some experimental results, including the RMSE (root mean square error) and R2 (R-Squared) metrics on both the training and testing sets, as well as the time cost for training each model. All models are trained using $k = 3$

Algorithm	Train		Test		Time
	RMSE	R2	RMSE	R2	
Gradient Boost Decision Tree (n=100, depth=5)	480002	0.6825	1349397	-3.818	1.55 Hour
Random Forest (n=200,depth=9)	471571	0.709	476664	0.399	24.33 Min
Random Forest (n=210,depth=8)	512009	0.657	482332	0.384	19.96 Min
Linear Regression(d=2)	510993	0.658	5237383	-72	1.75 Min
Linear Regression(d=3)	468162	0.713	97768842	-25293	13.73 Min
Decision Tree(depth=20)	270733	0.904	1538618	-5.26	26.95 Min

TABLE I: The performance of each model

cross-validation.

From the table, it can be observed that the decision tree easily trained a severely overfitting model (Note, this decision tree is trained using the features after PCA, while the decision tree mentioned above that is used to get the feature importance is trained using the features without PCA). In comparison, gradient boosting did not perform well on the training set, and it incurred significant time overhead. In contrast, linear regression with polynomial extension on features achieved a model with performance on the training set similar to gradient boosting but in much less time. However, as the polynomial extension degree increased, linear regression exhibited substantial errors on the test set. Among all models, random forest demonstrated a balanced performance with relatively low errors on both the training and test sets, with a reasonable time overhead, making it the most ideal choice comparatively.

VII. Conclusion And Future

In summary, this project use PySpark and Databricks to analyze a large dataset of UK housing properties. I explore the dataset, looking at trends and distributions, and build machine learning models to predict housing prices. The models I use include gradient boosting, random forest, and linear regression. I carefully adjust model settings, tune the hyperparameters, use cross-validation, and apply polynomial expansion to enhance our analysis.

Despite the challenges of working with big datasets, my results show that PySpark and Databricks are effective tools for handling large-scale data analytics. Among the models, random forest performs well on both training and test sets.

This research not only provides insights into the UK housing market but also demonstrates the practical use of distributed computing tools for handling big data challenges. The findings and methods used in this study offer a foundation for further exploration and improvement in data science and machine learning applied to real estate datasets.

Besides, this project also assists me in establishing a robust foundation for handling big data, the utilization of PySpark and Databricks proved instrumental in efficiently processing big data.

Furthermore, there are several avenues for further improvement in this project.

Firstly, the experiment results reveal that the performance of all the models to housing prices is not particularly high, both on the training and test sets. To enhance model accuracy, potential tasks include further tuning of hyperparameters for random forest and gradient boosting. Since predicting housing prices falls within the realm of Time Series, exploring the use of neural network architectures, such as LSTM or even transformer, may yield better performance in the context of large datasets.

Secondly, it's crucial to note that the dataset used in this project is a subset of the original 28 million rows, containing only 1.5 million rows. Exploring how to leverage the distributed and parallel computing capabilities of PySpark and Databricks (or other cloud platforms) to fully utilize the complete dataset is another promising direction for future research.

VIII. Reference

[1] PySpark Documentation. Apache Spark. https://spark.apache.org/docs/latest/api/python/

[2] Databricks Documentation. https://docs.databricks.com/en/getting-started/index.html

[3] Kaggle Dataset. https://www.kaggle.com/datasets/lorentzyeung/price-paid-data-202304