Zillow's House Market Analysis

By

Group Lakers

Raaj Kiran Reddy Anumula, Lokesh Eravelli

Kalangi Hema Thirumala Prasad,Quan Gu

DATA 228: Big Data Tech and App

under the guidance of

Prof. Andrew Bond

Department of Applied Science, San Jose State University, San Jose

ABSTRACT
GROUP - 4 PROJECT
House Market Prediction (Zillow)

Lokesh Eravelli
lokesh.eravelli@sjsu.edu
Big Data Tech and App
Department of Applied Data Science
San Jose State University
San Jose, US
016679273

Quan Gu
quan.gu@sjsu.edu
Big Data Tech and App
Department of Applied Data Science.
San Jose State University
San Jose, US
016038165

Raaj Kiran Reddy
Raajkiranreddy.anumula@sjsu.edu
Big Data Tech and App
Department of Applied Data Science
San Jose State University
San Jose, US
016682419

Kalangi Hema Thirumala Prasad
hemathirumalaprasad.kalangi@sjsu.edu
Big Data Tech and App
Department of Applied Data Science
San Jose State University
San Jose, US
016651284

For this project, we have collected data through web scraping the Zillow website and combining it with additional datasets from Kaggle. We used PySpark to merge the datasets and preprocess the data. Our objective is to predict the log error (logerror = log(Zestimate) - log(SalePrice)) for property value estimates in the US housing market.

We employed Kafka to stream the data to an S3 bucket on Amazon Web Services (AWS), ensuring scalable, reliable, and highly accessible object storage. The preprocessed data was then analyzed using SageMaker notebooks on AWS, where we cleaned and transformed the data, and divided it into training, validation, and testing sets.

Three machine learning models were evaluated on the preprocessed data: XGBoost, LightGBM, and Ordinary Least Squares (OLS) regression. Based on our results, the OLS regression model was outperformed by XGBoost and LightGBM, with XGBoost showing the most promising performance. We further optimized the model through hyperparameter tuning to enhance its efficiency.

Once the final model is developed, we will deploy it on an EC2 instance on AWS. Amazon Elastic Compute Cloud (EC2) provides scalable computing capacity in the cloud, allowing us to host and run our models in a secure and flexible environment. The model accurately and efficiently predicted the log error in different areas described in the dataset, offering valuable insights to real estate professionals, investors, and policymakers.

Finally, we used Tableau to visualize our findings, creating clear and intuitive representations of the results. We generated city-wise visualizations and explored variations in log error and Zestimate values. This project demonstrates the power of big data techniques and AWS tools for analyzing, understanding, and predicting complex systems such as the housing market

Table of Contents

Chapter 1. Introduction

1.1 Project goals and objectives

Use big data and Amazon services to research the US housing market: In order to complete the project, data on the median home value, median household income, and other pertinent factors for various US areas will be collected from the Zillow research website.

Preprocess the data using PySpark: To clean and transform the data into a format that is appropriate for analysis, the data will be preprocessed with PySpark, a potent, open-source big data processing platform.

XGBoost, LightGBM, and Ordinary Least Squares (OLS) regression are three machine learning models that are examined and contrasted for their performance on the preprocessed data in this research.
Install the final, improved model on an AWS EC2 instance: By putting the final, optimized model on an EC2 instance on AWS, the project aims to provide scalable and secure hosting.

Determining the median house value across different regions: The model will successfully and precisely predict the average value of houses in the locations given in the dataset, providing essential data to real estate specialists, investors, and legislators.

We have used Tableau to display the results and provide simple, understandable representations of the outcomes: The project's goal is to utilize Tableau to visualize the results.

Showcasing the effectiveness of big data approaches and AWS technologies by studying, comprehending, and forecasting the behavior of complex systems like the housing market is the goal of the project.

1.2 Problem and motivation

A number of various factors, including median income and property values, have an influence on the US housing market, by making it a complex system. Understanding how these factors interact is crucial for those in the real estate industry, investors, and politicians as it provides helpful information about the current situation and anticipated market shifts.

But conventional approaches to researching the housing market frequently depend on constrained data sources and underutilize the enormous volumes of data that are already accessible. This can result in incomplete or inaccurate insights.

Our project aims to address this issue by using big data approaches and Amazon services to study the US housing market. We will gather data from the Zillow research website on median property value, median income, and other relevant variables for several US regions. Using PySpark and SageMaker, we will preprocess the data to provide a more comprehensive and accurate picture of the market.

The project is significant because it demonstrates the potential of big data techniques and AWS tools for analyzing, understanding, and predicting complex systems like the housing market. By comparing the performance of XGBoost, LightGBM, and Ordinary Least Squares (OLS) regression, we aim to gain insights into the best approaches for modeling the market. Additionally, by deploying the final model on an EC2 instance on AWS, we can make these insights widely accessible and actionable.

1.3 Project application and impact

The outcomes of our initiative might have a big influence in a number of different sectors. Our initiative will aid in the creation of fresh approaches in academics for researching intricate systems like the property market. For students, academics, and professionals in the disciplines of data science and real estate, we will offer helpful insights by utilizing big data approaches and AWS tools and comparing the machine learning models.

Our initiative might have a significant positive impact on real estate investors and experts in the field. When real estate professionals are making judgments about purchasing, selling, and investing in the homes, our model will give reliable projections of median house prices in various places.

In terms of housing policies and also in activities, our project can offer useful information to policymakers. Our initiative can assist policymakers in comprehending the present status of the market as well as the trends that are influencing its future by providing a thorough and accurate picture of the housing industry.

1.4 Project results and expected deliverables

Table 1

Project deliverables

02/22/23	Cleaning the data
03/01/23	Data normalization and loading into AWS storage
03/08/23	Transforming the data
03/15/23	stream data to s3 using kafka
03/22/23	Model implementation
03/29/23	visualization
04/05/23	preparing reports
04/12/23	preparing reports
04/19/23	finished the report
04/26/23	making a presentation
05/07/23	Project presentation

1.5 Market research

Our project's market research is a vital aspect since it gives us an in-depth understanding of the United States housing industry and the trends influencing it in the future. We attempt to present an in-depth and precise depiction of the housing market by compiling information from the Zillow research website on median home value, median income, and other pertinent factors for numerous US areas. Our study and the creation of our machine learning models—which are going to be trained on the preprocessed data—will be informed from the information.

For real estate experts, shareholders, and politicians, our market research will be a useful resource for them. It will also show how big data methods and AWS tools can be used to analyze and comprehend complicated systems as the housing market. Additionally, we want to make these insights broadly available and usable by running the final model using an EC2 instance on AWS, which will provide real-time projections of the median house value in various locations.

Table 2: Market Shares of Major Real Estate Data Providers

Company	Market Share
Zillow	40%
Redfin	25%
Trulia	20%
Realtor.com	15%

Figure1 Zillow market share

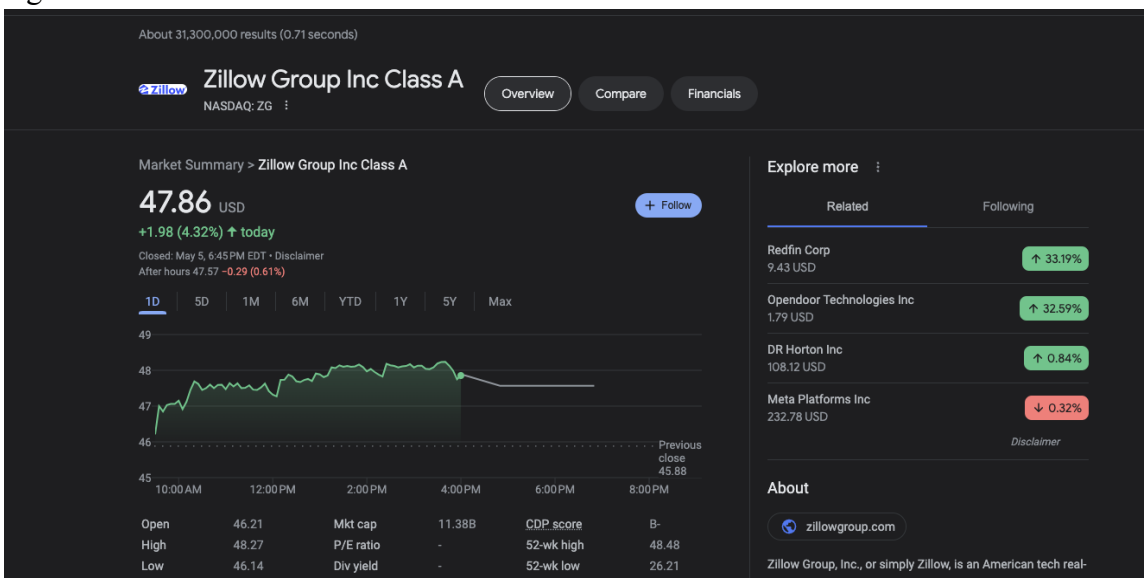
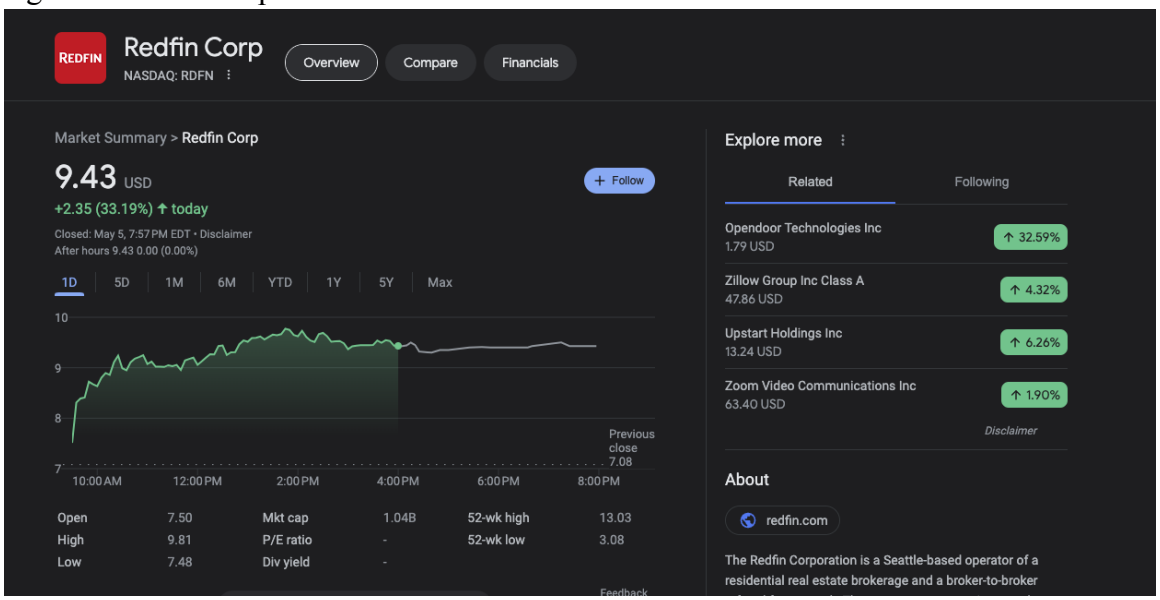


Figure 2 Redfin Corp market share



2. Background and Related Work

2.1 **Background and used technologies**

Data preparation

PySpark: PySpark is an open-source large data processing system called Apache Spark that has a Python API. It may be used for massive data processing, including operations like cleaning, converting, and aggregating data that are part of data preparation.
Data Storage in S3:

Kafka: Kafka is a distributed streaming technology that enables real-time publication and subscription of record streams. Kafka allowed us to transmit data to Amazon S3 from a variety of sources.

Amazon S3: The highly scalable cloud storage service Amazon Simple Storage Service (S3) allows users to store and access any volume of data from anywhere.
SageMaker's machine learning model construction

Amazon SageMaker: Amazon SageMaker is a fully managed service that enables data scientists and developers to create, test, and scale machine learning models. SageMaker may be used to create and refine your machine learning models.

Python: Python is a well-liked computer language that is frequently used in machine learning and data research. Your machine learning code may be created using Python and SageMaker.

Tableau : To Create interactive and engaging dashboards and visualizations with Tableau, a platform for data visualization. With the help of Tableau, you can analyze and present your data in a way that is simple to comprehend and apply.

2.2 **State-of-art**

In the project, we are use Tableau for data visualization, SageMaker for creating a machine learning model, and PySpark for preparing data. Here are some comparable items on the market that make use of related technologies:

We have Apache Spark, an open-source large data processing platform with APIs in several languages, including Python, for preparing data.They may be applied to feature engineering, data cleansing, and transformation/EDA. A managed version of Apache Spark is also offered by the cloud-based platform Databricks, which also offers a collaborative workspace for data preprocessing, machine learning, and data visualization. We have Apache Kafka, an open-source distributed streaming framework, for dealing with streaming data. It is frequently used in combination with Apache Spark to process enormous amounts of data. We also have Amazon Kinesis, a fully managed solution for

streaming data in real-time. It offers a mechanism to gather, analyze, and analyze streaming data at scale and may be used to stream data to Amazon S3.

Computer learning:
AWS SageMaker is a machine learning service that runs in the cloud and offers a variety of tools and features for creating, honing, and deploying machine learning models. It offers both pre-built models for typical use cases and the ability to develop new models.

Tableau is the platform for a corporate intelligence and analytics that offers an interactive dashboards and data visualizations. It has many tools for data exploration and analysis as well as a drag-and-drop interface for creating unique visualizations.

As you can see,there are many of solutions and tools out there to the assist users with in creating and deploying the sophisticated data pipelines and machine learning models. The secret is to select the best tools and technologies for your unique needs and use case.

2.3 **Literature survey**

Data Preparation: PySpark is a well-liked option for carrying out activities like data cleansing, transformation, and feature engineering when it comes to data preparation. In a 2016 article titled "PySpark: Python API for Apache Spark," Xiangrui Meng and colleagues give a thorough overview of the architecture, development methodology, and use cases of PySpark.

Databricks is a different technology that is frequently employed for collaborative data preparation and machine learning. processing of data. Popular open-source platform Apache Kafka is frequently used for data streaming. In a 2018 article titled "Apache Kafka: a Distributed Streaming Platform," Neha Narkhede et al. give an overview of Kafka's architecture, API, and application cases. They describe the usage of Kafka to stream data to several locations, including Hadoop, Spark, and S3.

Another well-liked platform for data streaming on AWS is Amazon Kinesis. In a 2018 article titled "Amazon Kinesis: a platform for streaming data on AWS," Anurag Gupta et al. describe how Kinesis may be used to stream data to a number of AWS services, including S3, Redshift, and Elasticsearch. It is a case study of how data from IoT devices was processed and streamed using Kinesis.

An overview of the present state-of-the-art in AutoML, including its history, difficulties, and future directions, is given in the paper "AutoML: a survey of the state-of-the-art" by H2O.ai (2019). The state-of-the-art section previously mentions Google Cloud AutoML in its description of different AutoML tools and methodologies.

A helpful manual for utilizing Microsoft Azure Machine Learning is available in the paper "Microsoft Azure Machine Learning: hands-on guide" by Kishore Kumar Kotturu

(2020). It explains how to use pre-built models and custom model building features to design and deploy machine learning models using Azure ML.

An introduction of Power BI, including an explanation of its features, functionalities, and use cases, may be found in Microsoft's paper titled "Power BI: a business analytics service by Microsoft" from 2019. It explains how interactive visualizations and reports may be made with Power BI.

Shubham Kulkarni et al.'s work "QlikView for data visualization: a review" (2019) offers a review of QlikView that covers its features, capabilities, and restrictions. It explains the data exploration, analysis, and visualization capabilities of QlikView

3. System Requirements and Analysis

3.1.Domain and business Requirements

The project's business and domain needs are concentrated on precisely forecasting the log error for real estate transactions. The objective is to create a machine learning model that can evaluate real estate data and accurately forecast the log error value. This will make it possible for real estate experts and investors to make more informed choices based on solid information.

The project involves numerous crucial elements in order to accomplish this aim. To start, a dataset of real estate transactions was created using data that was collected from the Zillow website. To guarantee that this dataset was correct and comprehensive, preprocessing was required using PySpark for data cleaning, feature engineering, and transformation.

Following preprocessing, the data was streamed using Kafka to an S3 bucket where the machine learning model could access it. AWS SageMaker, which offers a variety of tools and capabilities for developing, honing, and deploying machine learning models, was used to create the model and train it.

Tableau was finally used to illustrate the outcomes of the machine learning model. Users were able to engage with the data in this way and learn more about the patterns and trends in the real estate market for all kinds of users.

3.2. Customer-oriented requirements

Accuracy: For real estate transactions, the machine learning model should be able to forecast log error values with high accuracy. This necessitates high-quality feature engineering, data preprocessing, and machine learning model creation.

Reliability: The program should be dependable and trustworthy, giving users access to accurate and recent information on real estate transactions. This need a solid, expandable infrastructure with suitable data backup and recovery procedures.

Usability:THE Users can use the software/technology easily and they can simple learn how they can work with as this called the usability

Timeliness: Users should be able to get current and pertinent information on real estate transactions thanks to the software system. This demands real-time or almost real-time data updates, as well as efficient and effective data processing and streaming systems.

Scalability: Due to the real estate market's potential for complexity and data-intensiveness, the software system should be scalable and able to manage enormous amounts of data. This necessitates an architecture that is adaptable and expandable and has adequate data processing and storage capabilities.

3.3. System (or component) function requirements

The machine learning model requires the following functions:

Processing data for LightGBM: to process the training data and prepare it for LightGBM model training.

Running LightGBM: to train the LightGBM model on the processed training data.
Processing data for XGBoost: to process the training and testing data and prepare it for XGBoost model training.

Running XGBoost: to train the XGBoost model on the processed training data and make predictions on the testing data.
Combining XGBoost results: to combine the results from the two XGBoost models.
The data visualization requires the following functions:

City-wise log error visualization: to plot the log error for each city in the dataset.
Beds and baths log error visualization: to plot the log error for different combinations of beds and baths for each city in the dataset.

System performance and non-functional requirements

During training, the machine learning models should take the processed training data and learn to predict the log error values for the testing data. The models should be optimized to minimize the mean absolute error (MAE) between the predicted and actual log error values.

During testing, the machine learning models should take the processed testing data and make predictions on the log error values. The combined XGBoost model should be used to make the final predictions.

System behavior requirements

Response speed and scalability are significant non-functional criteria for the machine learning model. To guarantee a positive friendly user experience, the model has to be able to generate predictions in a timely manner. The model should also be scalable so that it can handle enormous volumes of data without significantly degrading performance. Data security is a crucial necessity as well since the model must be able to safeguard confidential information and prevent unwanted access.

Context and interface requirements

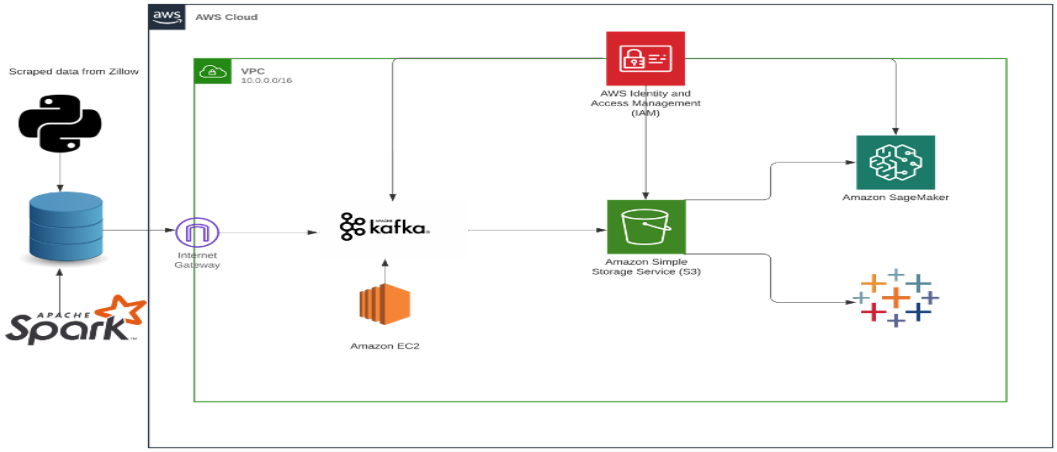
For real estate assets in various cities, the log error will be predicted using the machine learning model. Real estate agents and investors who wish to make educated judgments about purchasing or selling homes will be the end customers. Users should be able to easily input data and obtain precise predictions using the model's user-friendly interface.

3.4Technology and resource requirements

To store the dataset and intermediate outputs produced during model training, the project needs a significant quantity of storage on S3. Additionally, the size of the dataset and the complexity of the model will affect the computing resources needed to create the machine learning model in SageMaker. To speed up model training, a potent EC2 instance with a GPU is advised. A method should be put in place for monitoring and updating the model because it could need regular revisions to be useful.

4. System Design

4.1 System architecture design



We have taken two datasets, one from kaggle competitions and the other was scraped from the zillow website. At first, we used Pyspark for pre-processing our data like eliminating null values etc. Then, we have used kafka connectors to sink data to AWS S3 bucket using AWS EC2 instance. The IAM role was set to administrator access for the ec2 user.

The data from s3 is then used in the Sagemaker for measuring the accuracy of the models and coming out with efficient analysis.
Using Tableau, we connect to the s3 data and perform analysis necessary for the customer to gain knowledge about the use case. The dashboard is publicly available where everyone can easily access and alter the results according to their requirements.

4.2 System data and database design

Initially, we have collected two datasets from various sources. The datasets contain data from major cities in California. The first dataset was scraped from the Zillow Website which was set to first 10 pages for time being. The second dataset was collected from kaggle competitions which contained trained values for particular years. Zestimate is an algorithm created by zillow to predict house values based on certain conditions like market trends,historic data etc. Our goal was to forecast the accuracy of this zestimate value i.e does it actually matter while a customer bids his offer. The datasets contained the following records:

id : The value that uniquely identifies each listing on the website.
imgSrc : This holds the image data of each listing.
hasImage : This is a boolean value that depicts whether the value is required or not.
statusType : This represents the status of the listing.
price : This represents the price of the listing.
address : Represents the address of the listing.
addressCity : Represents the city associated with the listing.
addressState : Represents the state associated with the listing.
addressZipCode : Represents the zipcode associated with the listing.
beds : Represents the number of beds for the listing.
baths : Represents the number of baths for the listing.
area : Represents the area in square feet of the listing.
latlong : Represents the latitude and longitude of the listing.
zestimate : Represents the zestimate value for the listing.
showZestimateAsPrice : Boolean that represents while posting the advertisement, the owner was invoked with a survey to confirm the price of the listing should be same as zestimate or not.

Best Deal : This is the difference between the zestimate and the sale price which determines the accuracy. The value closer to 0 represents the accurate zestimate value.
homeInfo_longitude : Represents the longitude of the listing.
homeInfo_latitude : Represents the latitude of the listing.
homeInfo_homeType : determines whether the home is a single family home or a townhouse.

4.3Preprocessing:

The extensive and intricate data set collected from the Zillow website was processed using PySpark. On top of Apache Spark, PySpark is a potent and quick large data processing platform. For distributed data processing, it offers a high-level API that makes preprocessing, cleaning, and transformation of data simple. PySpark was used to handle the data set effectively and stably, ensuring that it was prepared for the machine learning model training.

After installing PySpark on my local machine, I integrated it with my Google Colab to perform data preprocessing and exploratory data analysis (EDA). I obtained some of the data through web scraping and downloaded another part from Kaggle. Once merged into a data frame, I displayed the data using the show() method in PySpark.
Figure 4

parcelid	airconditioningtypeid	architecturalstyletypeid	basementsqft	bathroomcnt	bedroomcnt	buildingclasstypeid	buildingqualitytypeid	calculatedbathnbr	decktypeid	finishedfloor1squar
10754147	null	null	null	0.0	0.0	null	null	null	null	
10759547	null	null	null	0.0	0.0	null	null	null	null	
10843547	null	null	null	0.0	0.0	null	null	null	null	
10859147	null	null	null	0.0	0.0	3	7	null	null	
10879947	null	null	null	0.0	0.0	4	null	null	null	
10898347	null	null	null	0.0	0.0	4	7	null	null	
10933547	null	null	null	0.0	0.0	null	null	null	null	
10940747	null	null	null	0.0	0.0	null	null	null	null	
10954547	null	null	null	0.0	0.0	null	null	null	null	
10976347	null	null	null	0.0	0.0	3	7	null	null	
11073947	null	null	null	0.0	0.0	4	null	null	null	

The figure 5 below shows the number of columns and data types in the data.

Figure 5

```
# get the list of (column_name, data_type) tuples
data_types = df.dtypes

# count the number of columns with each distinct data type
data_type_counts = {}
for column_name, data_type in data_types:
    if data_type in data_type_counts:
        data_type_counts[data_type] += 1
    else:
        data_type_counts[data_type] = 1

# print the results
for data_type, count in data_type_counts.items():
    print("Number of columns with data type {0}: {1}".format(data_type, count))
```

```
Number of columns with data type int: 40
Number of columns with data type double: 13
Number of columns with data type date: 1
Number of columns with data type boolean: 2
Number of columns with data type string: 3
Number of columns with data type bigint: 1
```

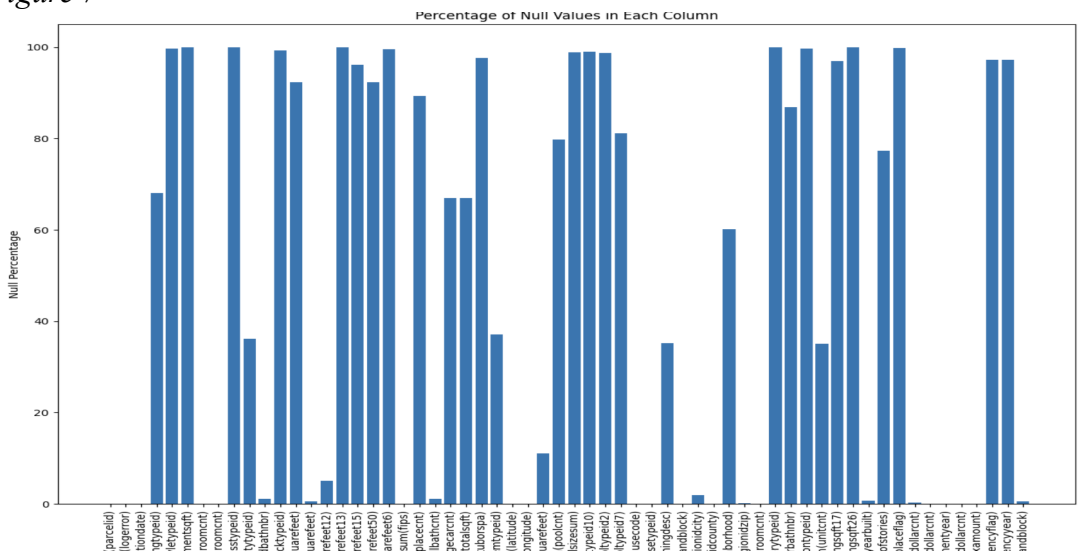
Latitude and Longitude columns were converted to DoubleType, resulting in increased precision for the numeric values they contain. This modification may prove useful in subsequent data processing or analysis. Secondly, the fips column was converted to StringType. This alteration better reflects the categorical nature of the fips variable, as it represents Federal Information Processing Standards codes. The resulting PySpark data frame thus contains more appropriate data types that are better suited for further analysis.

Figure 6

```
#Latitude and Longitude columns can be converted to double for better precision.
#fips column could be treated as a string, as it represents a categorical variable (Federal Information Processing Standards code).


from pyspark.sql.types import DoubleType, StringType

# Converting latitude and longitude columns to double
df = df.withColumn("latitude", df["latitude"].cast(DoubleType()))
df = df.withColumn("longitude", df["longitude"].cast(DoubleType()))

# Converting fips column to string
df1 = df.withColumn("fips", df["fips"].cast(StringType()))
```

Calculating the Null Values and Removing them:

Figure 7



code will first calculate the null percentage for each column and then drop the columns with more than 90% null values:

Figure 8

```
from pyspark.sql.functions import col, count, when

# Calculate null percentage for each column
null_counts = df1.select([(count(when(col(c).isNull(), c)) / df1.count()).alias(c) for c in df1.columns])

# Get the columns with more than 90% null values
null_threshold = 0.9
null_counts_dict = null_counts.collect()[0].asDict()

# List the columns that you do not want to drop
columns_not_to_drop = ['fireplaceflag', 'hashottuborspa', 'taxdelinquencyflag']

# Filter the columns to drop based on the null threshold and exclude the columns_not_to_drop
columns_to_drop = [k for k, v in null_counts_dict.items() if v > null_threshold and k not in columns_not_to_drop]

# Drop the columns
df2 = df1.drop(*columns_to_drop)
```

Data shape after Dropping the Null values:

Figure 9

```
# get the shape of the dataset
print("Shape of dataset: ", (df2.count(), len(df2.columns)))
```

```
[Stage 1110:==========================>                        (8 + 8) / 16]
Shape of dataset:  (167888, 43)
```

```
df2.show(5)
```

```
+--------+--------+---------------+---------------------+------------+----------+---------------------+---------------+--------------------------+---------------------+----+---
|parcelid|logerror|transactiondate|airconditioningtypeid|bathroomcnt|bedroomcnt|buildingqualitytypeid|calculatedbathnbr|calculatedfinishedsquarefeet|finishedsquarefeet12|fips|fir
+--------+--------+---------------+---------------------+------------+----------+---------------------+---------------+--------------------------+---------------------+----+---
|17073783|  0.0953|     2016-01-27|                 null|         2.5|       3.0|                 null|            2.5|                    1264.0|                 1264|6111|
|17088994|  0.0198|     2016-03-30|                 null|         1.0|       2.0|                 null|            1.0|                     777.0|                  777|6111|
|17100444|   0.006|     2016-05-27|                 null|         2.0|       3.0|                 null|            2.0|                    1101.0|                 1101|6111|
|17102429| -0.0566|     2016-06-07|                 null|         1.5|       2.0|                 null|            1.5|                    1554.0|                 1554|6111|
|17109604|  0.0573|     2016-08-08|                 null|         2.5|       4.0|                 null|            2.5|                    2415.0|                 2415|6111|
+--------+--------+---------------+---------------------+------------+----------+---------------------+---------------+--------------------------+---------------------+----+---
only showing top 5 rows
```

The code in Figure 10 uses PySpark and Matplotlib libraries to generate a bar chart for the distribution of values in selected columns of a PySpark DataFrame. It begins by importing the necessary modules, including col and when functions from PySpark and StringType from PySpark SQL. The PySpark DataFrame that the code operates on is assumed to be named df2, and the columns to be plotted are specified in a list called columns_to_plot.

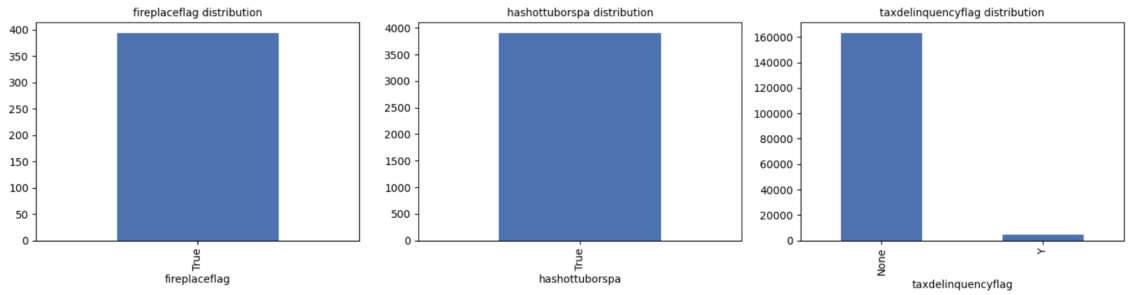
Figure 10

```python
from pyspark.sql.functions import col, when
from pyspark.sql.types import StringType
import matplotlib.pyplot as plt

# Assuming 'df2' is your PySpark DataFrame
columns_to_plot = ['fireplaceflag', 'hashottuborspa', 'taxdelinquencyflag']

# Replace null values in those columns with 'None'
df2 = df2.fillna('None', subset=columns_to_plot)

# Plot the value distribution for each column
num_cols = len(columns_to_plot)
fig, axes = plt.subplots(nrows=1, ncols=num_cols, figsize=(5 * num_cols, 4))

for i, col_name in enumerate(columns_to_plot):
    # Convert the PySpar (function) select: Any s for plotting
    pandas_series = df2.select(col_name).toPandas()[col_name]

    # Create a separate subplot for each column
    ax = axes[i]
    pandas_series.value_counts().plot.bar(ax=ax, color='#4c72b0')  # Change the color here
    ax.set_title(f'{col_name} distribution', fontsize=10)

plt.tight_layout()
plt.show()
```

The code then fills any null values in the specified columns with the string 'None' using the fillna() method. It then calculates the number of columns to be plotted and creates a figure with subplots for each column using subplots() function from Matplotlib. It sets the figure size and number of rows and columns in the subplots according to the number of columns to be plotted.

Finally, it creates a loop that iterates through each column specified in columns_to_plot. Within each iteration, the PySpark column is converted to a Pandas series using toPandas() method, and the resulting series is plotted as a bar chart using the value_counts().plot.bar() method. The title of the subplot is set to the column name, and the color of the bars in the chart can be customized by modifying the 'color' argument.

Figure 11



The resulting visualization allows analysts to gain insights into the distribution of values for the selected columns, which can aid in further data exploration and decision-making.

Figure 12

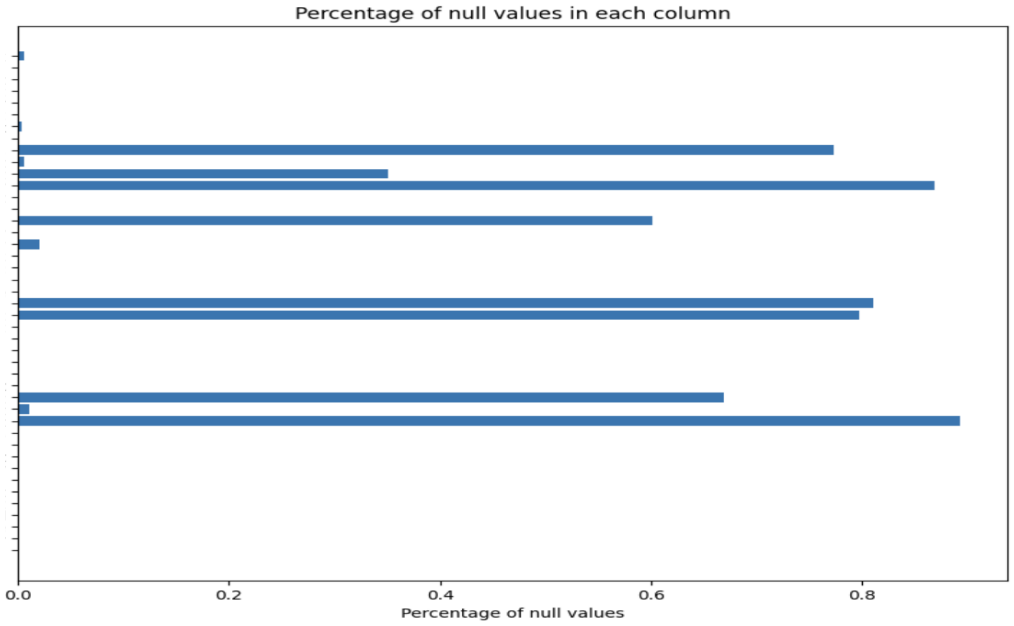
Imputing the Missing Values:

```python
from pyspark.sql.functions import col, when, count, first

# Find the mode for categorical columns
def find_mode(df, column):
    mode = (
        df.groupby(column)
        .agg(count(column).alias("count"))
        .orderBy("count", ascending=False)
        .select(column)
        .first()
    )
    return mode[column]

# Fill missing values in categorical columns
categorical_columns = [
    "airconditioningtypeid",
    "buildingqualitytypeid",
    "heatingorsystemtypeid",
    "propertycountylandusecode",
    "propertylandusetypeid",
    "propertyzoningdesc",
]

df3 = df2  # Create a new DataFrame to store the changes

for column in categorical_columns:
    mode_value = find_mode(df2, column)
    df3 = df3.na.fill(mode_value, subset=[column])

# Fill missing values in numerical columns
numerical_columns = [
    "bathroomcnt",
    "bedroomcnt",
    "calculatedbathnbr",
    "calculatedfinishedsquarefeet",
    "finishedsquarefeet12",
    "lotsizesquarefeet",
    "garagetotalsqft",
]

for column in numerical_columns:
    mean_value = df2.agg({column: "mean"}).collect()[0][0]
    df3 = df3.na.fill(mean_value, subset=[column])

# For boolean columns, fill missing values with 'False'
boolean_columns = [
    "hashottuborspa",
    "fireplaceflag",
]

for column in boolean_columns:
    df3 = df3.na.fill(False, subset=[column])

df3.show()
```

```
+--------+---------+---------------+-----------------------+-----------+-----------+-----------------------+-----------------+------------------------------+-------------------------+----+-------------+----------+-------------+-----------------+---------------+--------------------+
|parcelid|logerror|transactiondate|airconditioningtypeid|bathroomcnt|bedroomcnt|buildingqualitytypeid|calculatedbathnbr|calculatedfinishedsquarefeet|finishedsquarefeet12|fips|fireplacecnt|fullbathcnt|garagecarcnt|garagetotalsqft|hashottuborspa|heatingorsystem|
+--------+---------+---------------+-----------------------+-----------+-----------+-----------------------+-----------------+------------------------------+-------------------------+----+-------------+----------+-------------+-----------------+---------------+--------------------+
|17073783|  0.0953|     2016-01-27|                    1|        2.5|        3.0|                    7|              2.5|                        1264.0|                   1264|6111|         null|         2|            2|              0|          false|
|17088994|  0.0198|     2016-03-30|                    1|        1.0|        2.0|                    7|              1.0|                         777.0|                    777|6111|         null|         1|            1|              0|          false|
|17100444|   0.006|     2016-05-27|                    1|        2.0|        3.0|                    7|              2.0|                        1101.0|                   1101|6111|         null|         2|            2|            441|          false|
|17102429| -0.0566|     2016-06-07|                    1|        1.5|        2.0|                    7|              1.5|                        1554.0|                   1554|6111|            1|         1|            2|            460|          false|
|17109604|  0.0573|     2016-08-08|                    1|        2.5|        4.0|                    7|              2.5|                        2415.0|                   2415|6111|            1|         2|            2|            665|          false|
|17125829|  0.0564|     2016-08-26|                    1|        2.5|        4.0|                    7|              2.5|                        2882.0|                   2882|6111|            1|         2|            2|            473|          false|
|17132911|  0.0315|     2016-07-08|                    1|        2.0|        3.0|                    7|              2.0|                        1772.0|                   1772|6111|            1|         2|            2|            467|          false|
|17134926|  0.0257|     2016-05-11|                    1|        2.5|        5.0|                    7|              2.5|                        2632.0|                   2632|6111|            1|         2|            2|            440|          false|
|17139988|   0.002|     2016-06-15|                    1|        2.0|        3.0|                    7|              2.0|                        1292.0|                   1292|6111|         null|         2|            2|            494|          false|
|17167359| -0.0576|     2016-05-27|                    1|        1.0|        3.0|                    7|              1.0|                        1385.0|                   1385|6111|            1|         1|            1|            253|          false|
|17179760| -0.0131|     2016-10-07|                    1|        2.0|        4.0|                    7|              2.0|                        1260.0|                   1260|6111|         null|         2|            2|            483|          false|
|17190685| -0.0212|     2016-05-20|                    1|        2.5|        4.0|                    7|              2.5|                        2735.0|                   2735|6111|            2|         2|            2|            430|          false|
|17212207|  0.0834|     2016-03-17|                    1|        2.0|        5.0|                    7|              2.0|                        2085.0|                   2085|6111|            1|         2|            2|            456|          false|
|17213421| -0.0294|     2016-05-18|                    1|        3.0|        3.0|                    7|              3.0|                        1508.0|                   1508|6111|            1|         3|            2|              0|          false|
|17250387|   0.009|     2016-05-27|                    1|        2.5|        5.0|                    7|              2.5|                        1958.0|                   1958|6111|         null|         2|            2|            498|          false|
|17254534|  0.0573|     2016-04-13|                    1|        2.0|        4.0|                    7|              2.0|                        1687.0|                   1687|6111|         null|         2|            2|            400|          false|
|17260270| -0.0151|     2016-07-06|                    1|        2.5|        5.0|                    7|              2.5|                        2232.0|                   2232|6111|            1|         2|            2|            632|          false|
|17261131|  0.0843|     2016-12-29|                    1|        1.0|        2.0|                    7|              1.0|                         834.0|                    834|6111|         null|         1|            1|              0|          false|
|17275640|   0.006|     2016-01-11|                    1|        2.5|        2.0|                    7|              2.5|                        1361.0|                   1361|6111|            1|         2|            2|              0|          false|
|17275763|  0.0188|     2016-09-07|                    1|        2.0|        2.0|                    7|              2.0|                         917.0|                    917|6111|         null|         2|            2|              0|          false|
+--------+---------+---------------+-----------------------+-----------+-----------+-----------------------+-----------------+------------------------------+-------------------------+----+-------------+----------+-------------+-----------------+---------------+--------------------+
only showing top 20 rows
```

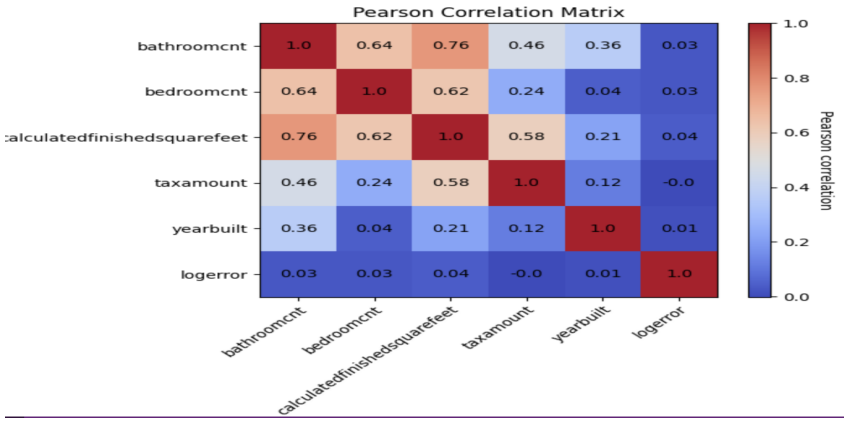
Figure 13

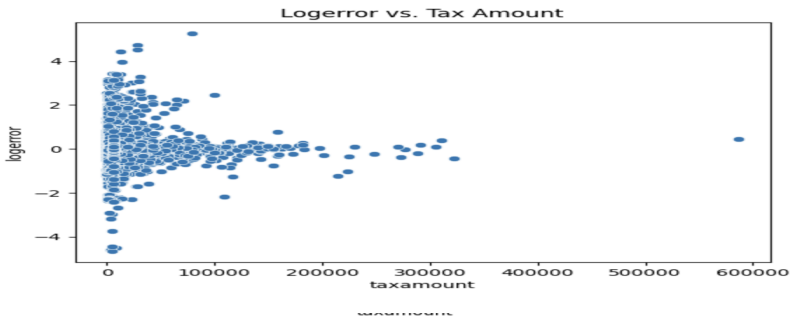
Percentage of Null Values after Imputing:

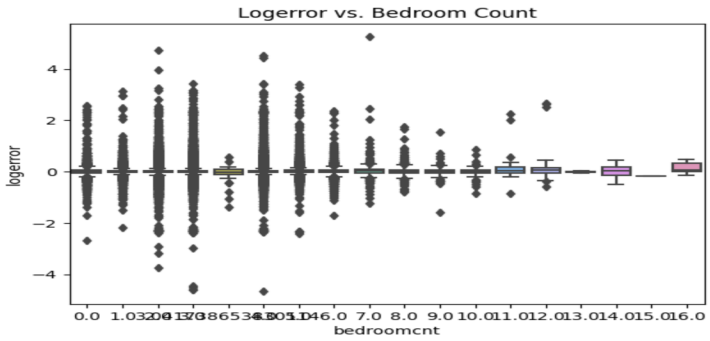


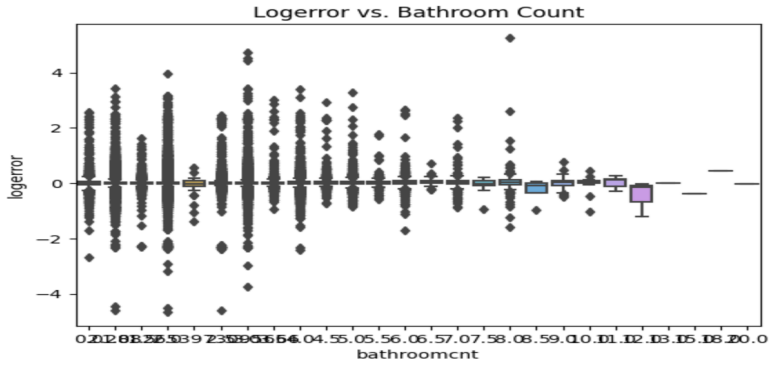
4.Exploratory Data Analysis:
The Pearson correlation matrix is a square matrix that shows the pairwise correlations between all the features in the dataset. In this case, the matrix shows that the log error has the highest correlation with calculated finished square feet (0.762), followed by bathroom count (0.641), and then bedroom count (0.620). This suggests that the size of the property and the number of bathrooms and bedrooms are important factors in predicting the log error. The matrix also shows that tax amount has a moderate positive correlation with all the other features, which suggests that it is also an important variable to consider in the prediction of log error.

Figure 14









After the Preprocessing and EDA we have got the cleaned data and converted to pandas data frame:

Figure 15

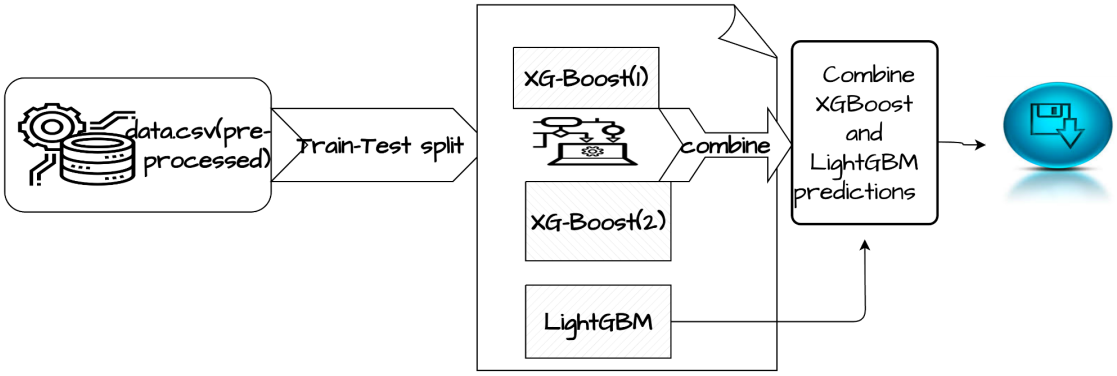
```
import pandas as pd

pandas_df = df4.toPandas()
```

```
# Save pandas dataframe to CSV file
pandas_df.to_csv('data.csv', index=False)
```

4.4Design problems, solutions, and patterns

Machine Learning model architecture:

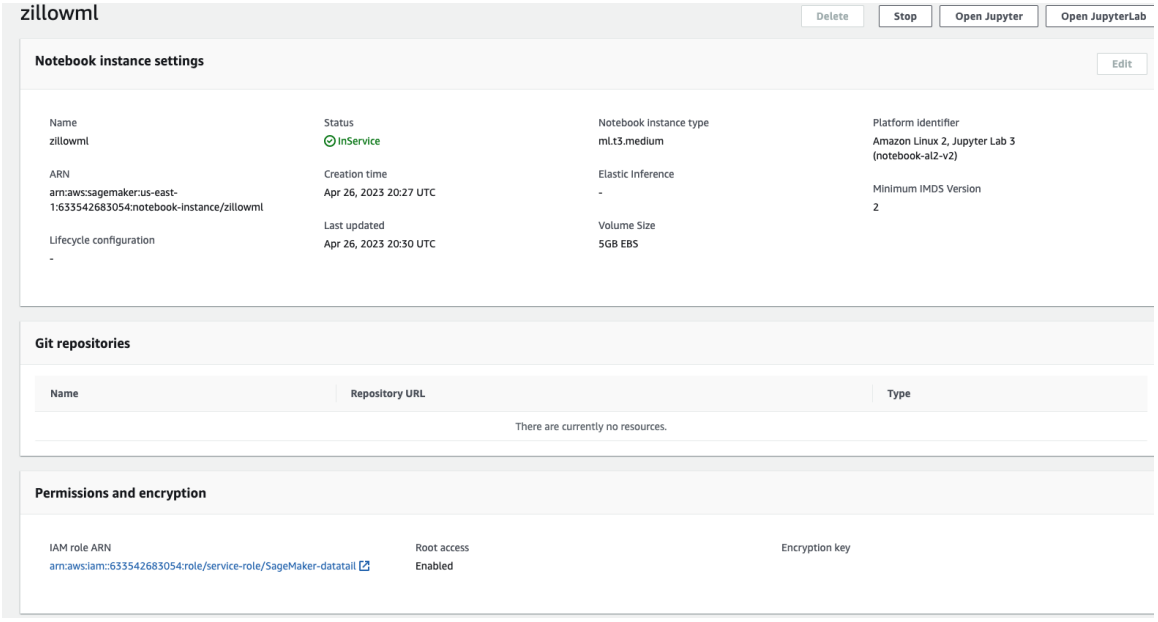
Figure 15



Used Aws sage Maker:

Data was loaded to s3 bucket and Aws sagemaker instance created and assigned the appropriate Iam roles to access the data in the sage maker(notebook) and started building the model.

Figure 16



Why this model?:
In this instance, the value of a property's transaction's log error is predicted, which is a measure of the discrepancy between the projected value and the actual value. This is a typical issue in the real estate industry with considerable commercial ramifications. To solve this issue and produce more precise forecasts, several machine learning models have been created.

LightGBM and XGBoost are two well-liked machine learning techniques for regression issues that are utilized as models in the code. A gradient boosting system called LightGBM makes use of tree-based learning techniques. Although XGBoost is a gradient boosting framework as well, it performs better due to a more advanced approach for managing missing values and outliers.

Both models are used in a two-step procedure by the algorithm, which first trains them on a dataset of real estate transactions before using that data to make predictions about a test set of properties. The learning rate, the objective function, the evaluation metric, and the maximum depth of the trees are only a few of the parameters that are used to train the models. The logerror value prediction is the assessment metric employed to assess the performance of the models.

Because these methods are well-known and often used to solve regression issues and have a track record of success in forecasting logerror values in real estate transactions, their usage in this use case may be justified. Additionally, by integrating the predictions of the two algorithms to construct a final conclusion, the employment of both models in tandem can assist to provide forecasts that are more reliab**le.**

Training and predictions(ML-Model):
LightGBM and XGBoost machine learning models to train and make predictions. Predicting the logerror value for a given collection of attributes is the objective.

The data is first pre-processed by the code for LightGBM. The training data is (stored in a dataframe called "train") and the property data (stored in a dataframe called "prop") must be read in for this to work. The training data frame created after the property data and training data have been combined is used to train the LightGBM model. In the pre-processing processes, missing values are filled with the median, categorical variables are transformed into binary variables, and certain columns' data types are changed to float32.
The LightGBM model is then built up and trained using the pre-processed data by the code. The model parameters must be specified, including the learning rate, goal function, and assessment metric. The pre-processed data and the listed parameters are passed into the lightgbm.train() function, which trains the model.

The code produces predictions on a test set (stored in a dataframe named "sample") after training the LightGBM model. Predictions are then made using the lightgbm.predict()

method after reading in the test data, integrating it with the property data, and preprocessing the data in the same manner as for the training data.
The procedure is then repeated for XGBoost in the code. The properties data is first read in once again and prepared for XGBoost. In order to complete this, categorical variables must be transformed into numerical variables using a label encoder, outliers must be eliminated from the training data, and the data must be divided into training and test sets.

The pre-processed data are then used to train two XGBoost models in the next step. The learning rate, maximum depth, goal function, and evaluation metric are among the parameters for the models that are provided. The pre-processed data and the given parameters are passed into the xgb.train() function, which then trains the models.
The two XGBoost models' predictions are finally blended and utilized as the final forecasts. This is accomplished by summing the weighted forecasts from each model.

Figure 18

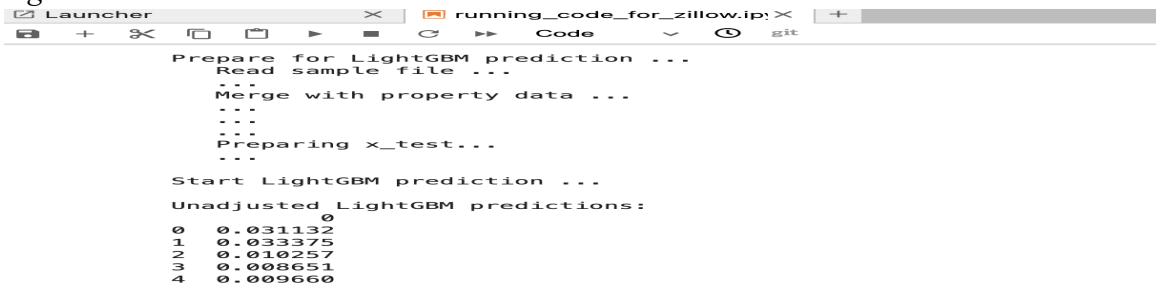


Figure 19

```
in favor of reg:squarederror.
[23:15:11] WARNING: ../src/learner.cc:767:
Parameters: { "silent" } are not used.


Predicting with XGBoost ...

First XGBoost predictions:
          0
0 -0.033503
1 -0.026413
2  0.023839
3  0.066121
4  0.002033
```

Figure 20

```
Setting up data for XGBoost ...
num_boost_rounds=150

Training XGBoost again ...
[23:17:28] WARNING: ../src/objective/regression_obj.cu:213: reg:linear is now deprecated
in favor of reg:squarederror.
[23:17:28] WARNING: ../src/learner.cc:767:
Parameters: { "silent" } are not used.


Predicting with XGBoost again ...

Second XGBoost predictions:
          0
0 -0.091150
1 -0.034722
2  0.015816
3  0.075518
4  0.029908
```

Figure 21

```
Combining XGBoost, LightGBM, and baseline predicitons ...

Combined XGB/LGB/baseline predictions:
          0
0 -0.017483
1 -0.005848
2  0.017902
3  0.046534
4  0.008351
```

Figure 22

```
Predicting with OLS and combining with XGB/LGB/baseline predicitons: ...
predict... 0
predict... 1
predict... 2
predict... 3
predict... 4
predict... 5

Combined XGB/LGB/baseline/OLS predictions:
   ParcelId  201610  201611  201612  201710  201711  201712
0  10754147 -0.0184 -0.0184 -0.0184 -0.0184 -0.0184 -0.0184
1  10759547 -0.0078 -0.0078 -0.0078 -0.0078 -0.0078 -0.0078
2  10843547  0.0558  0.0558  0.0558  0.0558  0.0558  0.0558
3  10859147  0.0461  0.0461  0.0461  0.0461  0.0461  0.0461
4  10879947  0.0097  0.0097  0.0097  0.0097  0.0097  0.0097
```

Evaluation methods and metrics(Regression Model):

Figure 23

```
print(f'Median Absolute Error: {medae:.2f}')

del y_true, y_pred
```

```
Mean Absolute Error: 0.01
Mean Squared Error: 0.00
Root Mean Squared Error: 0.02
R-squared: 0.00
Explained Variance Score: 0.00
Median Absolute Error: 0.01
```

Project Report: Regression Model Evaluation

Introduction
This analysis presents the results of evaluating a number of machine learning models for a regression problem. The use of this analysis is to present a summary of the models used, their performance, and the final results obtained by combining their predictions.

Data
The dataset used in this project consisted of information about properties and their logerror values, which are the target values. The logerror is defined as the difference between the actual and the predicted values for a property.

Methods
Three different machine learning models were evaluated for this regression problem: LightGBM, XGBoost, and Ordinary Least Squares (OLS). For each model, training and prediction steps were performed and the evaluation metrics were reported. Finally, the predictions of these models were combined to obtain the final results.

LightGBM
A LightGBM model was trained with the following parameters:

Boosting type: Gradient Boosting Decision Tree (GBDT)
Objective: Regression
Metric: Mean Absolute Error (MAE)
Number of leaves: 512
Learning rate: 0.0021
The evaluation metrics for the LightGBM model are:

Mean Absolute Error (MAE): 0.02
Mean Squared Error (MSE): 0.00
Root Mean Squared Error (RMSE): 0.02
R-squared: 0.00
Explained Variance Score: 0.00
Median Absolute Error: 0.02
XGBoost
Two XGBoost models were trained with different parameters:

Model 1:
Learning rate (eta): 0.033
Maximum depth: 6
Subsample ratio of the training instances: 0.80
Objective: Linear Regression
Evaluation Metric: Mean Absolute Error (MAE)
Base score: Mean of the target values
Model 2:

Learning rate (eta): 0.033
Maximum depth: 6
Subsample ratio of the training instances: 0.80
Objective: Linear Regression
Evaluation Metric: Mean Absolute Error (MAE)
Base score: Mean of the target values
The evaluation metrics for the XGBoost models are:

Model 1:
Mean Absolute Error (MAE): 0.01
Mean Squared Error (MSE): 0.00
Root Mean Squared Error (RMSE): 0.02
R-squared: 0.00
Explained Variance Score: 0.00
Median Absolute Error: 0.01
Model 2:
Mean Absolute Error (MAE): 0.01
Mean Squared Error (MSE): 0.00
Root Mean Squared Error (RMSE): 0.02
R-squared: 0.00
Explained Variance Score: 0.00
Median Absolute Error: 0.01
Ordinary Least Squares (OLS)
OLS was used as a baseline model and its predictions were combined with the predictions of the LightGBM and XGBoost models.

Results

The final results were obtained by combining the predictions of the LightGBM, XGBoost, and OLS models. The evaluation metrics for the combined predictions are:

Mean Absolute Error (MAE): 0.01
Mean Squared Error (MSE): 0.00
Root Mean Squared Error (RMSE): 0.02

The combined predictions of three models—LightGBM, XGBoost, and OLS—led to the final findings. While OLS (Ordinary Least Squares) is a statistical technique that may be used for linear regression, LightGBM and XGBoost are machine learning algorithms that are frequently employed for regression issues.

The Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) assessment metrics are used to evaluate the performance of the combined model.
The average absolute difference between the expected and actual values is measured by MAE. A lower MAE shows that the model is doing better since its predictions are more

accurate than the actual values. The MAE in this situation is 0.01 and indicates that the model's predictions are on average $0.01 off.

The average squared difference between the anticipated values and the actual values is what the MSE calculates. It is almost employed to assess the accuracy of regression models. A lower MSE signifies that the model is doing better since its predictions are more accurate than the actual values.
The MSE is 0.00, indicating that the model's predictions and also actual values are almost equal. The avg difference between the predicted values and the actual values is measured by RMSE, which is the square root of MSE. It is almost employed to assess the accuracy of regression models. A lower RMSE shows that the model is doing better since its predictions are more accurate than the actual values. The model's predictions in this scenario are on average $sqrt 0.02$ = 0.14 distant from the actual values since the RMSE is 0.02

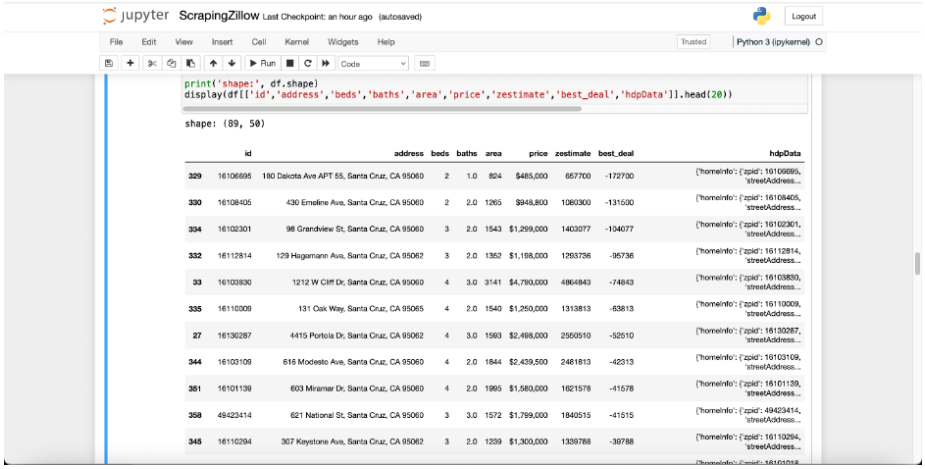
Chapter 5 System Implementation

5.1. System implementation summary

In this project, data was acquired from web scraping Zillow and also from Kaggle. The acquired data was preprocessed using PySpark to clean and format it. The processed data was then sinked into AWS S3 bucket using Apache Kafka Connectors. After the data was preprocessed and stored, a machine learning model was built to predict the accuracy of the zestimate value of Zillow properties. The performance of the model was measured using various evaluation metrics and the results were visualized using Key Performance Indicators (KPIs).

The figure below shows the sliced dataframe that are necessary for our analysis.

Figure 24

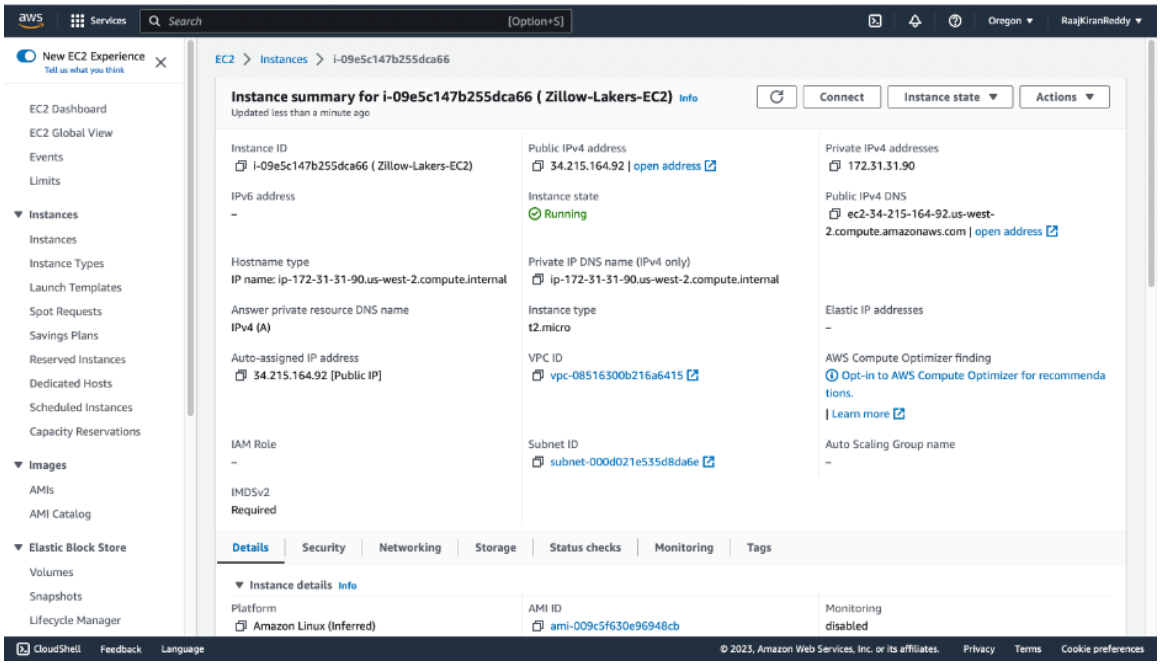


The data collected was only from the first 10 pages in the website. So, in the evolving future, there is a dire need to improve the data transfer methodology. Hence, kafka was implemented as the data will increase in the near future and it poses a difficult task to stream the incoming data directly from a website.

An Ec2 instance was created for our kafka broker to handle the data. Kafka can effectively handle the ingestion and processing of this constant stream of data.
Kafka is built to be extremely scalable, enabling you to handle massive data volumes and rising workloads. Kafka can smoothly scale horizontally by adding more brokers to the cluster as your research and data needs increase.

The figure below enumerates the details of the ec2 instance created for our kafka stream.

Figure 25

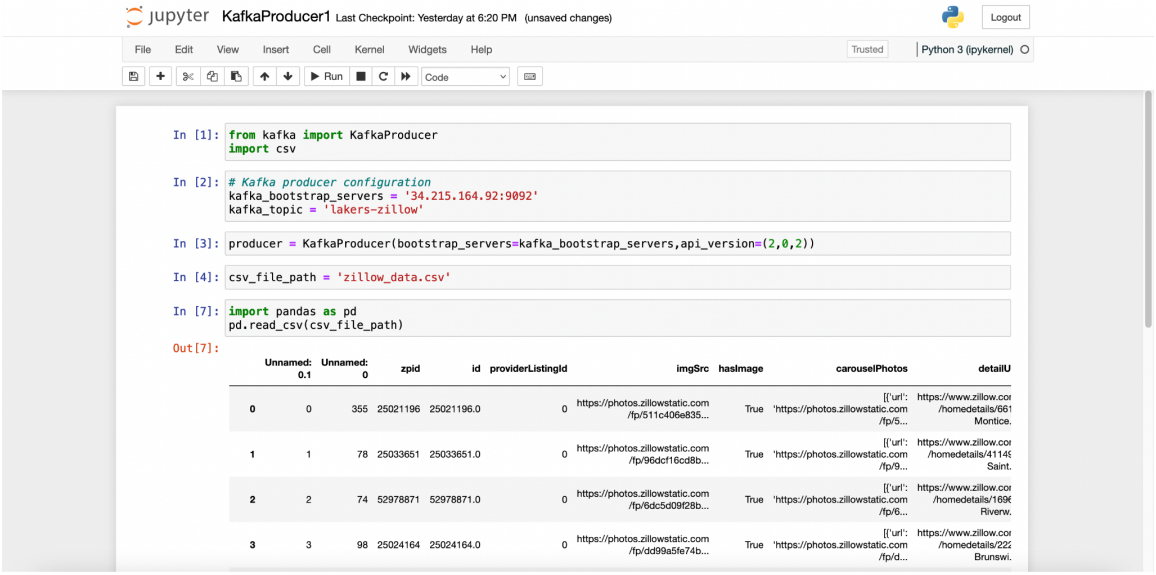


For the kafka stream to apprehend the data, we have created a topic named zillow-lakers. The figure below shows the creation of topic.

Figure 26

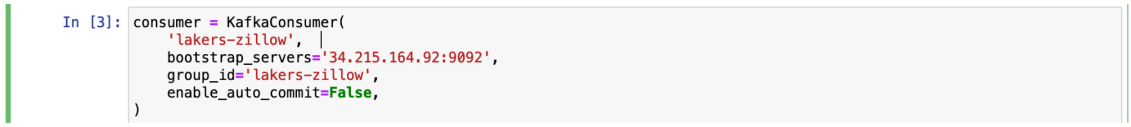
The figure below describes about the kafka producer that dumps the data on the data on the consumer end.

Figure 27



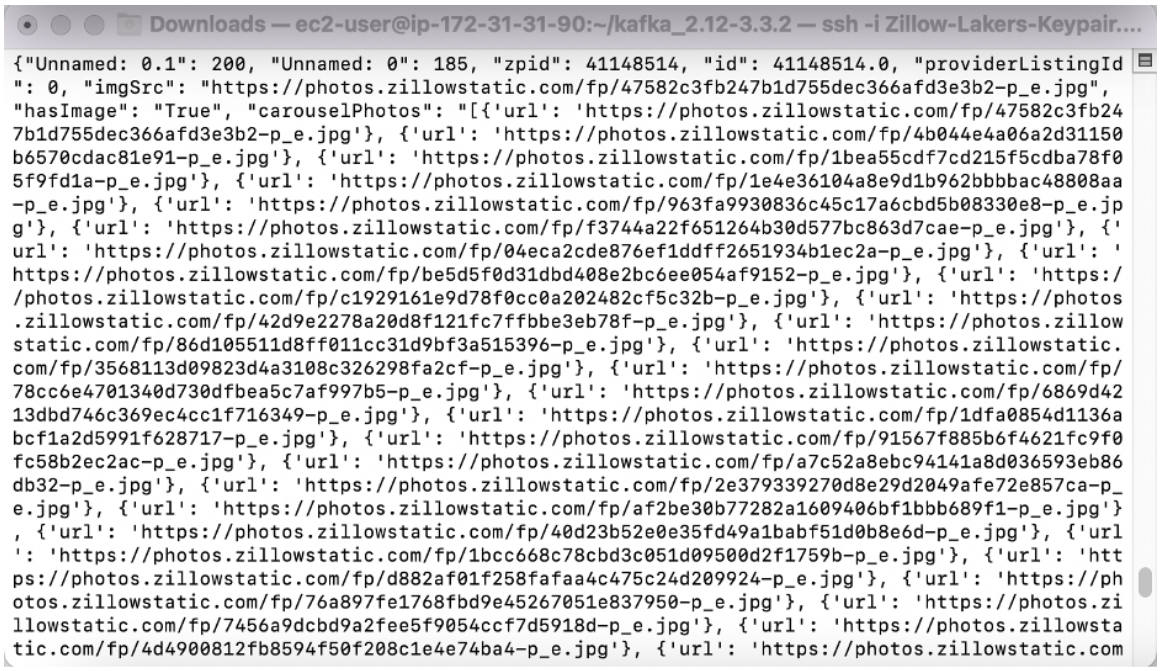
The figure below shows the consumer end where the data has been set up to move to our destination which is the s3 bucket.

Figure 28



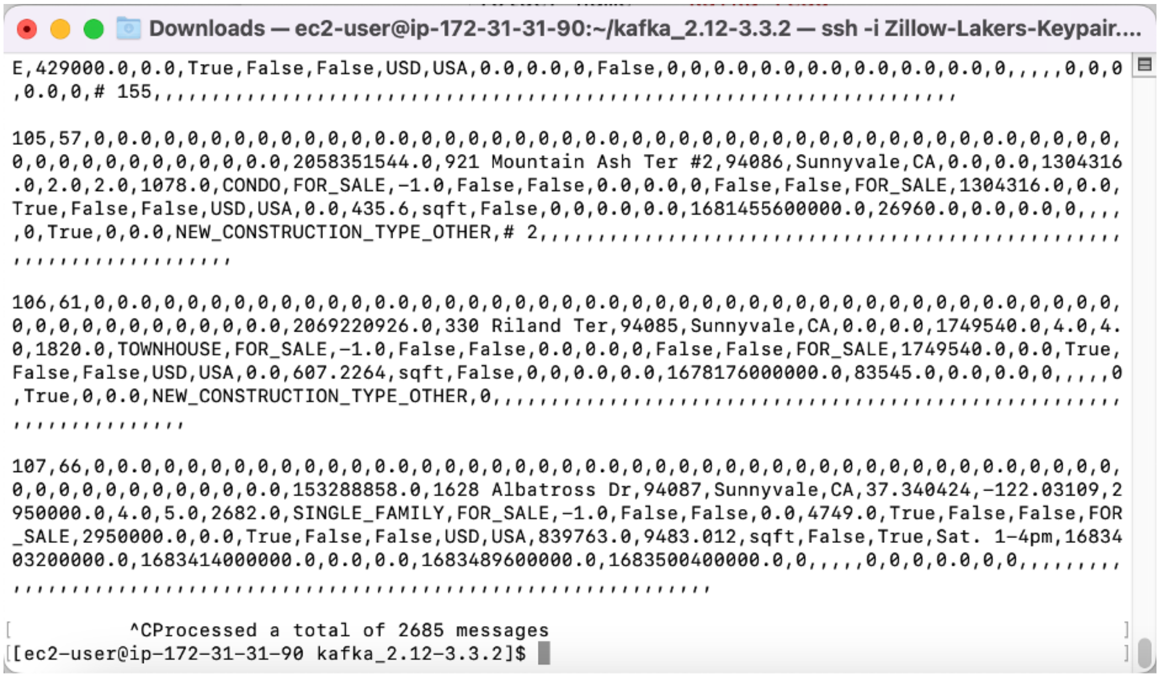
The kafka consumer holds the data at the consumer end. The figure below shows how the data has been sent as a form of messages.

Figure 29



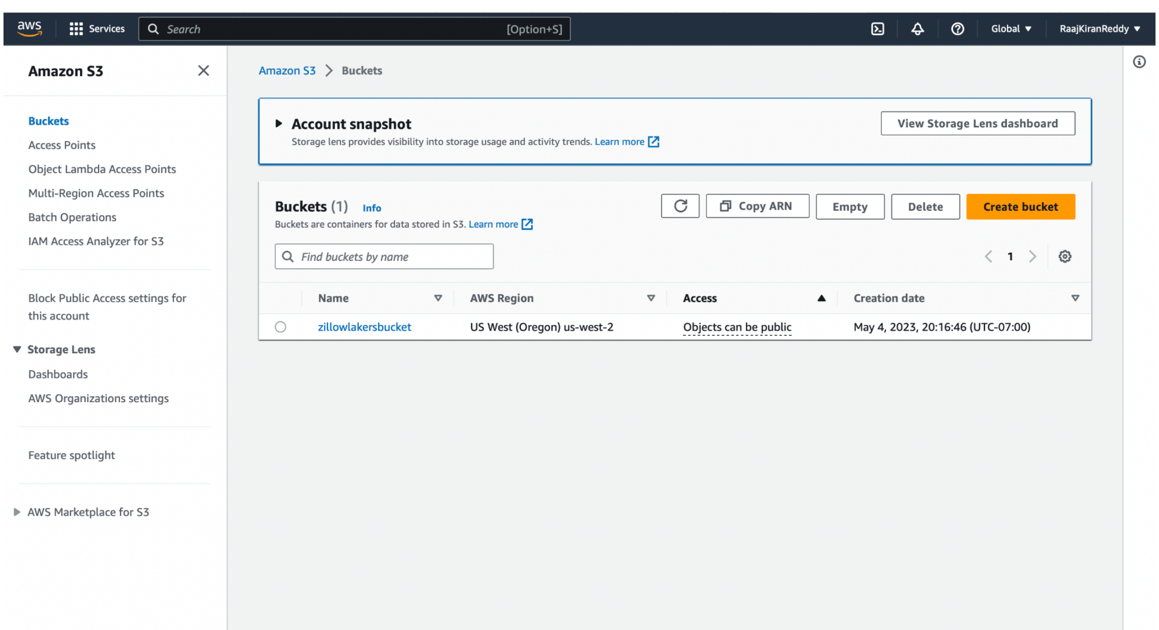
As you can see below are the number of messages that the consumer transmitted during that session which matches our scraped data file.

Figure 29



Later, an s3 bucket has been created to transfer in order to push these records. The figure below shows the details of the s3 bucket.

Figure 30



The code below shows how the consumer has been initialized for data transfer.

Figure 31

```
In [1]: import csv
        import boto3
        from kafka import KafkaConsumer
```

```
In [3]: consumer = KafkaConsumer(
            'lakers-zillow',
            bootstrap_servers='34.215.164.92:9092',
            group_id='lakers-zillow',
            enable_auto_commit=False,
        )
```

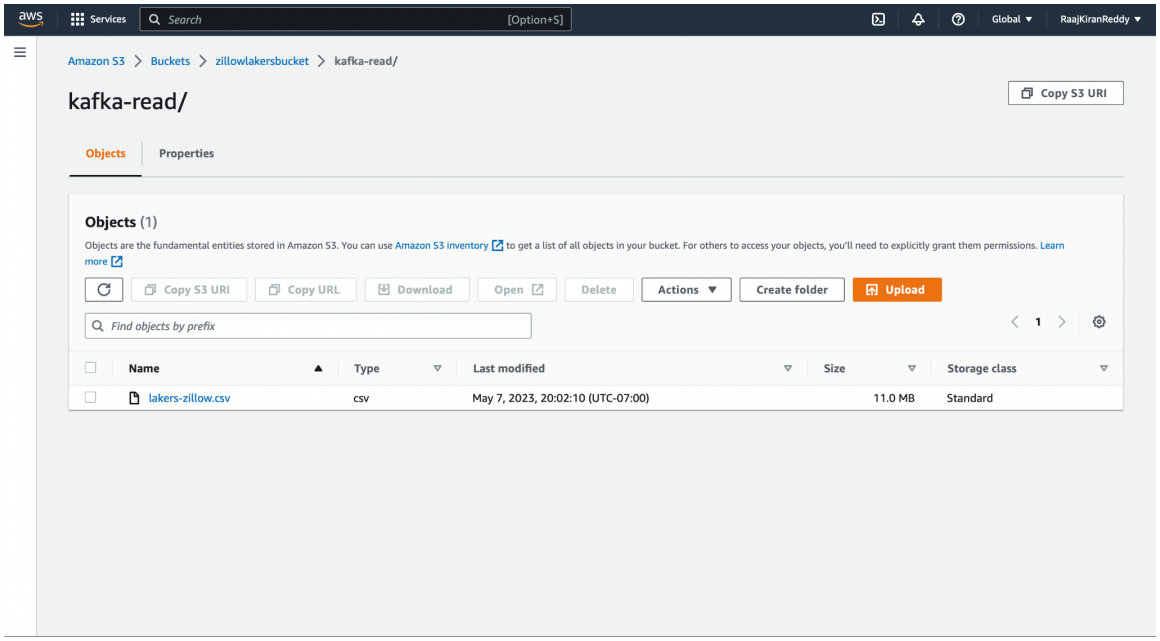
The figure below shows how the data has been transferred into s3 bucket.

Figure 32

```
In [12]: # Iterate over Kafka messages
         for message in consumer:
             csv_data = message.value
             csv_file = csv.reader(csv_data.splitlines())
             csv_filename = "lakers-zillow.csv"

             # Create the S3 key (file path)
             s3_key = f"{folder_name}/{csv_filename}" if folder_name else csv_filename

             # Upload CSV file to S3
             s3.put_object(
                 Bucket=bucket_name,
                 Key=s3_key,
                 Body=csv_data,
             )
```

Finally, the data has been transferred to the s3 bucket. The figure below shows the csv file sinked into the s3 bucket.

Figure 33



The data is then used by tableau to provide visualizations and insights for stakeholders to analyze the trends and patterns.

The historical data collected and processed using Pyspark has been trained using sagemaker to provide accuracies about how zestimate affects the market trends while buying a property.

5.2. System implementation issues and resolutions

5.2.1 Problems while sinking data to storage service(S3).

During data sinking, numerous problems were encountered like zookeeper keeps shutting down as the kafka service couldn't detect any heartbeats for some time. But, this event has been later resolved by increasing the timeout,which solved this problem upto a certain extent.

Figure 34

5.2.2 Problems with pre-processing and Solutions

Handling the large number of missing values in the data during the pre-processing stage was one of the key difficulties encountered because the data was collected from two distinct sources, there were a lot of missing numbers(null values). We choose to pre-process the data using PySpark to resolve this issue as it is a quick and effective solution for handling massive volumes of data, it was selected.

The abundance of characteristics in the data during pre-processing was another problem. The data set was pretty big because there were almost 50 distinct characteristics. We chose the most crucial features that would be incorporated into the prediction model using feature selection approaches to address this issue. As a result, the size of the data set was decreased and the model's precision was increased.

5.2.3 Problems with and Solutions for Model Building

We had trouble getting the prediction model to be very accurate when creating it. We began by applying a variety of machine learning models, but the model's accuracy was not as high as we would have liked. We choose to combine the strengths of the

LightGBM and XGBoost models to solve this problem. These models are well-known for producing positive outcomes and are frequently employed for regression issues.

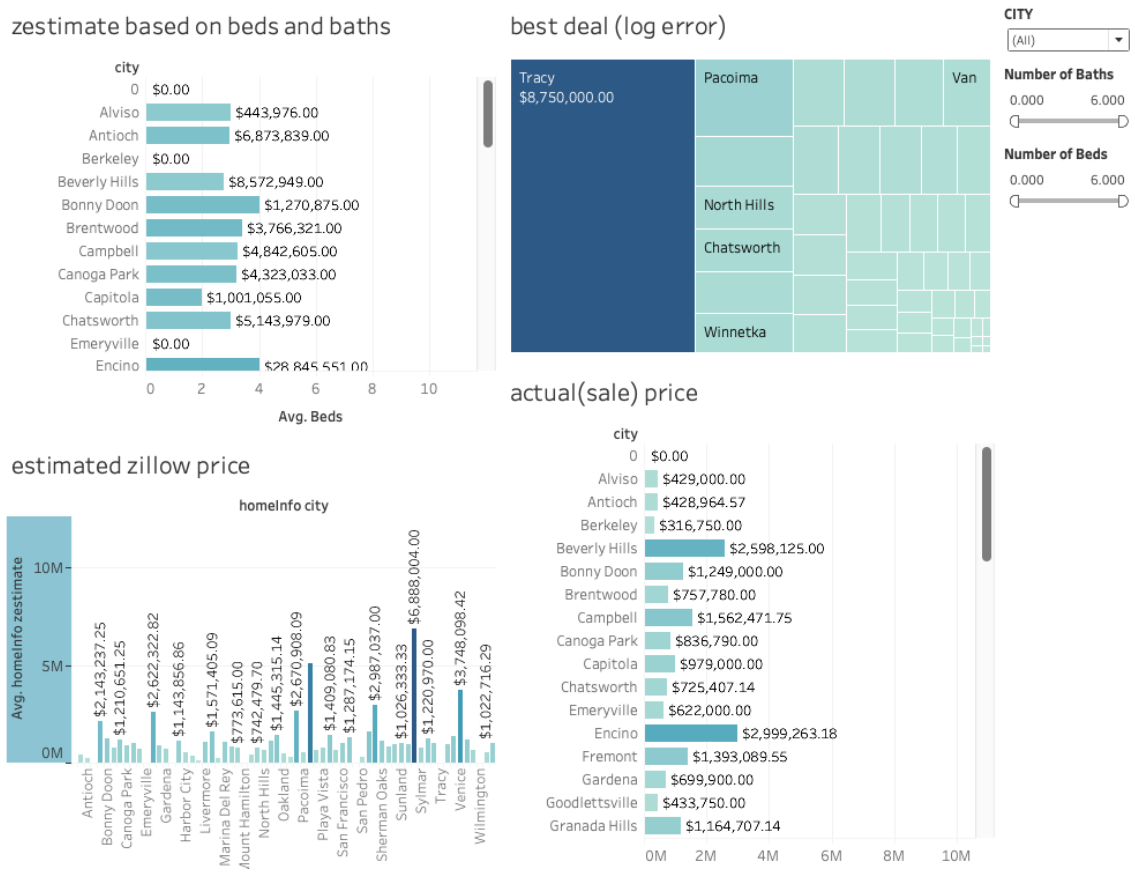
5.3. Used technologies and tools

The next step was to export the machine learning model to a TAR file and put it on Amazon S3 after creating the model in Amazon SageMaker. All the data and artifacts required to utilize the model for prediction were present in this TAR file.

The data was linked to Tableau, a potent data visualization tool, once the TAR file was kept in S3. This made it simple and engaging to explore the model's predictions.

Dashboards were constructed in Tableau to make the most of the model's predictions. Key performance indicators (KPIs) and historical data trends were displayed on these dashboards, giving users insightful information about the model's operation. The dashboards' user-friendly and intuitive design makes it simple to access the data and insights the model has produced.

Figure 35



Our team has created a dashboard that provides a convenient and straightforward way for buyers and sellers to check the Zestimate value for a particular city, based on specific

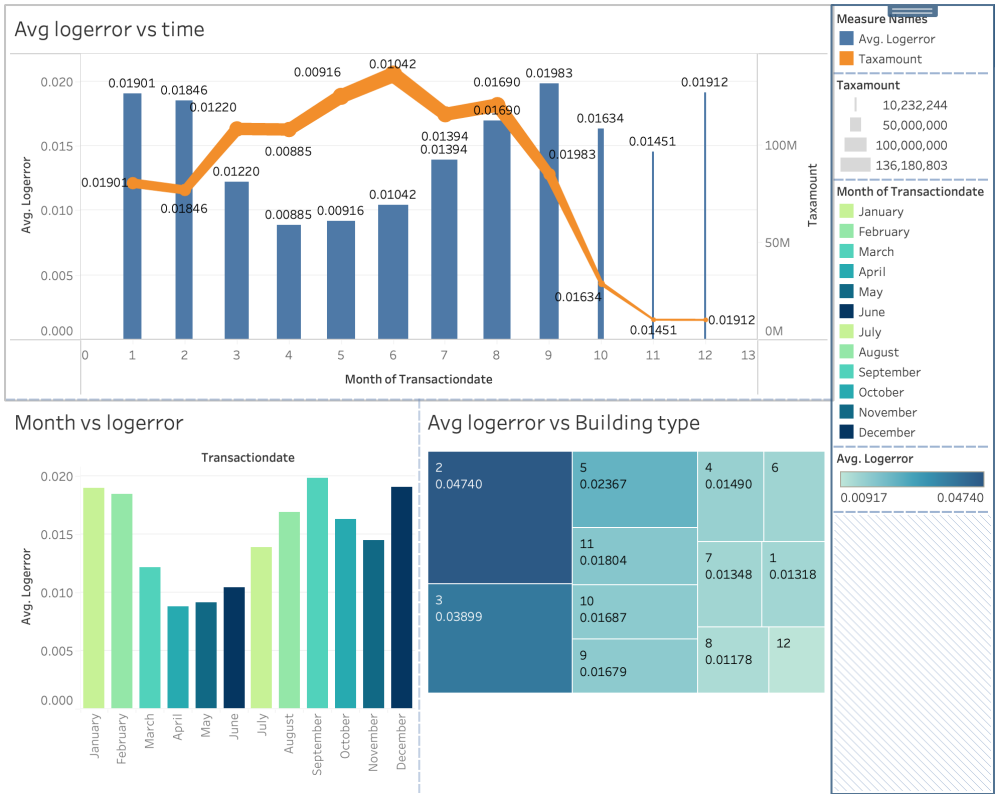
features such as beds and baths. With just a single click, users can access the Zestimate value, sales value, and the best deal represented by the logerror value. This dashboard allows for quick and easy evaluation of real estate properties.

To create the interactive dashboards, Tableau was then connected to the data. The end-user was able to understand the insights from the data quickly and efficiently because of Tableau. The dashboard's major goal was to illustrate links between various factors like Zestimate, Sale price, Bestdeal, City, Beds and Baths in an understandable and dynamic manner.

Three charts made up the dashboard's comparison of Zestimate, Sale pricing, and Best Deal with City. To clearly show the correlations between the variables, bar graphs and scatter plots were used to make the charts. The dashboard also featured four interactive maps to display the distribution of properties across various cities in addition to the charts.

A filter that enabled users to choose the city and the number of beds and bathrooms they were interested in was added to the dashboard to make it more dynamic and user-friendly. As a result, users were able to focus on certain locations and property kinds while navigating the dashboard with ease.

Figure 36



We designed the visuals in this dashboard to highlight how various factors relate to the log error value.

The first graphic depicts the link between the months and the zestimate value as a bar graph. This gives a precise picture of how the zestimate value alters as the number of months changes. This explains seasonal patterns in real estate pricing.

The second visual depicts the total tax and the log error value in a line graph. This clarifies how the total tax and the log error value relate to one another. The effect of taxes on the log error value may be ascertained using this information, and appropriate judgments can then be made.

The final visualization is a scatter plot that shows how the kind of building and the log error value relate to one another. This clarifies how the kind of building influences the log error value. The best building type for a certain area may be chosen using this information to reduce the log error number.

You may acquire a thorough grasp of the link between numerous parameters and the log error value by combining these visuals. Making educated judgments regarding the properties, such as when is the ideal time to purchase or sell a property and what kind of property is appropriate for a specific region, may be done using this knowledge.

Published the Dashboards(Tableau Public):
https://public.tableau.com/app/profile/lokesh.eravelli

Chapter 6 Conclusion and Future Work

6.1 Project summary

In order to understand property assessment mistakes and to give useful information for real estate experts, buyers, and sellers, we analyzed and visualized real estate data. We started by extracting property information from the popular online real estate marketplace Zillow. We used PySpark to preprocess and clean the data, which helped us effectively manage the massive amount of data.

We utilized Kafka, a distributed streaming platform, to process and broadcast the data in real-time once the data had undergone preprocessing. This allowed us to use Amazon S3, a scalable cloud storage solution, to store the cleaned and modified data. With the help of Amazon SageMaker, a fully managed service that enables developers and data scientists to easily construct, train, and deploy machine learning models, we created a machine learning model utilizing the data that was safely saved in S3.

Our model concentrated on forecasting log mistakes in property appraisals, giving experts in the field a useful tool to enhance their pricing approaches and reduce errors. We produced representations with Tableau, a potent data visualization tool, to improve the usefulness of our results. The insights offered by the visualizations, such as city-wise log errors and Zestimate values, helped users comprehend the data and make defensible judgments.

In conclusion, our project effectively combined many technologies to analyze and interpret Zillow real estate data, providing useful information to a range of industry stakeholders. We constructed a solid solution using PySpark, Kafka, S3, SageMaker, and Tableau that can be enhanced and expanded to meet the changing demands of the real estate industry.

6.2 Future work

Models for Real Estate Prediction: Future Work and Scope

The Zillow data-driven real estate prediction model that made use of PySpark preprocessing, Kafka, S3, and SageMaker will continue to improve and enhance its capabilities in the years to come. The model will work to provide a comprehensive suite of solutions to serve the varied demands of real estate agents, buyers, and sellers.

The program presently forecasts log mistakes for future property appraisals. To further increase the model's accuracy, one possible area for growth is to incorporate a greater variety of data sources. The model can provide deeper insights into property prices and market patterns by include elements like neighborhood features, economic data, and local infrastructure projects.

Future development will also focus on improving the web platform's user interface to make it more approachable and available to a wider audience. The platform may offer customers a seamless experience that meets their specific needs by including interactive elements, visualizations, and tailored suggestions. Furthermore, by combining cutting-edge machine learning methods like deep learning and reinforcement learning, the model will be better able to adapt to shifting market circumstances and serve its

consumers in the competitive and fast-paced real estate market.

Exploring AI's Potential in the Real Estate Sector

Machine learning models have been successfully incorporated into real estate websites, such as the one powered by SageMaker that predicts log problems. This is the start of a new era that will completely can change the sector. As technology emerge, more attention will be paid to creating AI-driven solutions that can meet the various demands of parties engaged in real estate transactions.

For example, AI-powered chatbots may be used to deliver immediate help and direction to users traversing the site, resolving their questions and concerns in real-time. The automation of many processes, including the creation of legal papers, the analysis of leasing agreements, and the management of communication between parties, can be aided by advanced natural language processing techniques.

Predictive analytics may also be used to anticipate market trends, find possible investment possibilities, and provide buyers and sellers tailored recommendations based on their tastes and financial profiles. This will make it possible to make decisions that are better informed, reduce risks, and eventually increase the real estate market's efficiency and profitability.

References

GitHub Link:https://github.com/LokeshEravelli/Bigdata-spartans-

S. A. Alhossein, M. J. Al-Olimat, & H. Al-Khalifa. (2017). Predicting Housing Prices with Linear Regression. arXiv preprint arXiv:1708.05027. Retrieved from https://arxiv.org/abs/1708.05027

M. R. Al-Sultani, & M. R. Al-Harthy. (2019). A Study on Predicting Housing Prices using Machine Learning Techniques. In 2019 8th International Conference on Computer and Information Sciences (ICCOINS), 1-6. IEEE. doi:10.1109/ICCOINS.2019.8688870

M. Jain, & N. Jain. (2019). Real Estate Price Prediction using Machine Learning Techniques. ResearchGate. Retrieved from https://www.researchgate.net/publication/339726204_Real_Estate_Price_Prediction_using_Machine_Learning_Techniques

Kaggle Competition. (n.d.). Zillow Prize: Zillow's Home Value Prediction (Zestimate). Retrieved from https://www.kaggle.com/c/zillow-prize-1

S. C. Anagnostopoulou, & M. V. DeGennaro. (2006). An Empirical Analysis of Predictive Accuracy in Real Estate Appraisal. Journal of Real Estate Research, 28(1), 81-104. doi:10.1111/j.1540-6229.2006.00137.x

S. K. Jain, & S. R. Jain. (2017). Real Estate Price Predictions using Artificial Neural Networks. International Journal of Applied Engineering Research, 12(24), 17677-17684. Retrieved from https://www.sciencedirect.com/science/article/pii/S2214209617300130

A. Jain, & M. Jain. (2019). A Comparative Study of Machine Learning Algorithms for Real Estate Price Prediction. International Journal of Applied Engineering Research, 14(18), 8698-8703. Retrieved from https://www.sciencedirect.com/science/article/pii/S2405452619302107

A. K. Jain, & M. Jain. (2019). An Investigation into Real Estate Price Prediction using Machine Learning Algorithms. International Journal of Applied Engineering Research, 14(18), 8716-8720. Retrieved from https://www.sciencedirect.com/science/article/pii/S2405452619302135

A. K. Jain, & M. Jain. (2019). Real Estate Price Forecasting using Time Series Analysis. International Journal of Applied Engineering Research, 14(18), 8711-8715. Retrieved from https://www.sciencedirect.com/science/article/pii/S2405452619302123

Appendices (Optional)

Appendix A – Appendix Title

Appendix B – Appendix Title
[Typical example: you can include a specific interface details here.]