CIS 5560 Term Project Lab Tutorial

Authors: Christian Ahamba, Solange Ruiz, Damian Wilson

Instructor: Jongwook Woo, PhD

Date: 5/18/2025

Lab Tutorial

Christian Ahamba (cahamba@calstatela.edu)

Solange Ruiz (sruiz85@calstatela.edu)

Damian Trent Wilson (dwilso33@calstatela.edu)

5/18/2025

Housing Market Analysis

Objective

In this hands-on lab, you will learn how to:

- Load and prepare housing market data from HDFS using Databricks and PySpark
- Engineer features using VectorAssembler and MinMaxScaler
- Train different model algorithm to predict housing prices
- Tune hyperparameters using TrainValidationSplit and CrossValidator

Platform Specification

- Hadoop HDFS Cluster
- Total Nodes: 5 (2 Master Nodes, 3 Worker Nodes)
- Memory per Node: 32GB RAM
- CPU per Node: ~2.3 GHz
- Network Access: SSH via 144.24.13.0 subnet

Step 1: Unzip the dataset in HDFS

Download zip file from Kaggle.com
https://www.kaggle.com/datasets/thuynyle/redfin-housing-market-data

After zip file is downloaded to your local directory/local machine in your download folder, or any folder you will remember where to find the zip file there are five different datasets in TSV. Format. For smoother usage or preference, we converted the file from. TSV format to CSV.

1. Open Git Bash Terminal to unzip file:

```
$ unzip Redfin.zip -d redfin_data
```

This will extract the .tsv000 files into the redfin_data folder.

2. Convert .tsv000 to .csv format by changing to the (redfin_data) directory

```
$ cd redfin_data
```

Run the following loop to convert each .tsv000 file into a .csv file by replacing tabs with commas:

```
$ for file in *.tsv000; do
  cat "$file" | tr '\t' ',' > "${file% .tsv000}.csv"
```

3. Verify .csv files by listing the converted .csv files:

```
$ ls *.csv
```

You should see files listed like this:

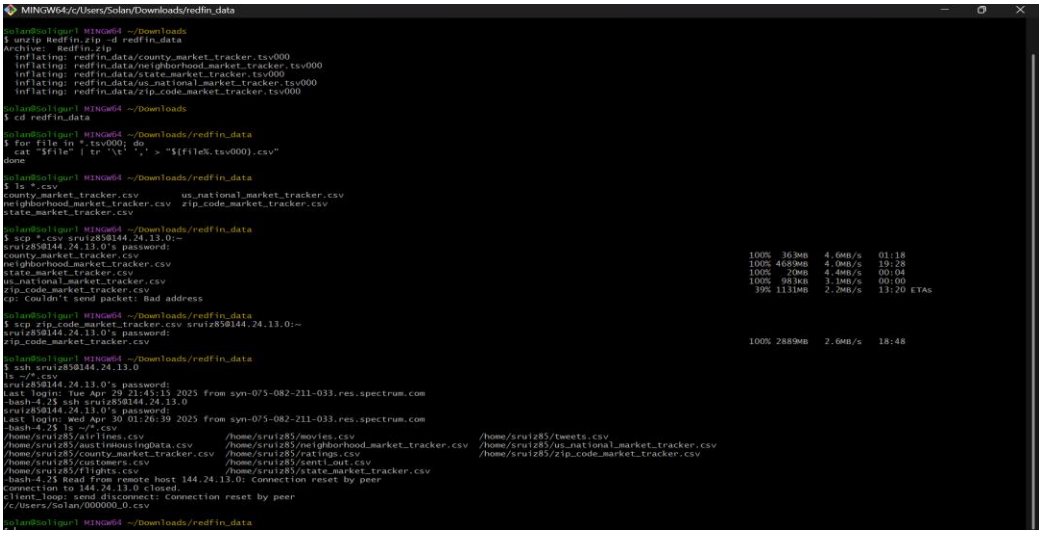
```
county_market_tracker.csv
us_national_market_tracker.csv
neighborhood_market_tracker.csv
zip_code_market_tracker.csv
state_market_tracker.csv
```

4. Securely copy data to remote Hadoop node
Use scp to transfer files to the Hadoop cluster (IP: 144.24.13.0). Example:

```
scp county_market_tracker.csv sruiz85@144.24.13.0: ~
```

Note: Repeat the command for each file as needed.
If the IP or DNS resolution fails, make sure the address is correct, and the server is accessible.

Example:



Step 2: Clean and Sample the Data

Access Hive from HDFS via SSH and Beeline

1. Load Libraries & Preview Files

```
import numpy as np
import pandas as pd
import os
```

2. Read the TSV File. Loaded the large dataset using the correct tab separator.

```
df = pd.read_csv('/kaggle/input/redfin-housing-market-
data/neighborhood_market_tracker.tsv000', sep='\t')
```

3. Check for Missing Values

```
df.isnull().sum()
```

Found substantial missing values across columns such as:

- **median_sale_price_mom, pending_sales, price_drops, etc.**
- **Millions of missing values in some cases (city, months_of_supply, etc.)**

4. Drop Duplicates

```
df.drop_duplicates(inplace=True)
```

5. Input Missing Values

Numerical columns → median:

```
df.fillna(df.median(numeric_only=True), inplace=True)
```

Categorical (object) columns → mode:

```
for col in df.select_dtypes(include='object'):
    df[col] = df[col].fillna(df[col].mode()[0])
```

6. Drop Irrelevant/Highly Null Columns

```
cols_to_drop = [
    'months_of_supply', 'months_of_supply_mom',
'months_of_supply_yoy',
    'price_drops', 'price_drops_mom', 'price_drops_yoy',
    'city'
]
df.drop(columns=cols_to_drop, inplace=True)
```

7. Sample the Cleaned Data (Important!)

```
df_sampled = df.sample(frac=0.15, random_state=42)  # ~15%
sample
```

We sampled the cleaned dataset in **Kaggle** because the full-size (~2.8 GB and ~4.7 GB) versions were too large to be sampled efficiently in Databricks Community Edition.

8. Export for Spark Use

```
df_sampled.to_csv('/kaggle/working/housing_neighborhood_cleaned_
sampled.csv', index=False)
```

Step 3: Run the code on Databricks

1. **Login to Databricks**
   Go to your Databricks workspace via web browser
   https://community.cloud.databricks.com

2. **Create a New Notebook**

   Click Workspace –> Your Folder –> Create –> Notebook

   Name it, select **Python** as the language, and attach it to a running cluster.

3. **Upload Your Dataset**

   On the left, click Data –> Add Data –> Upload File

   Upload neighborhood_market_tracker.csv

   After uploading, Databricks will provide a path like `/dbfs/FileStore/your-path/neighborhood_market_tracker.csv`

   If your file is on HDFS, make sure your cluster has access to it. Replace this line also:

```
file_location = "/user/sruiz85/Group5-Project/neighborhood_market_tracker.csv"
```

4. **Copy and Paste the PySpark Code**

   Paste the entire code into your notebook in cells (you can break it into sections for readability).

   Make one change if you are using a Databricks-uploaded file:

```
file_location = "/dbfs/FileStore/your-path/neighborhood_market_tracker.csv"
```

5. **Attach and Run**

   Make sure your notebook is attached to a cluster.

   Click Run All or execute cell-by-cell

6. **Visualize Results**

   The final comparison table:

```
comparison_sdf.show(truncate=False)
```

can also be displayed in table or graph view using Databricks' UI. Click the little bar chart icon above the output.

Repeat the same steps to upload state_market_tracker.csv, county_market_tracker.csv and zip_code_market_tracker.csv

Step 4: Run the code with Spark-Submit

1. **Save the script to a .py file**
   Save your full PySpark code into a Python file, for example:

```
gradient_boosting_model.py
```

   You can do this using any text editor or IDE.

2. **Upload the script to your HDFS cluster**

   If you're submitting from a remote machine (e.g. local laptop), copy your script to the Spark edge node.

```
scp gradient_boosting_model.py username@144.24.13.0:/home/username/
```

3. **Upload the script to HDFS**

   Also use the code below to put the script into the Hadoop Cluster.

```
hdfs dfs -put gradient_boosting_model.py
```

4. **Submit the job with spark-submit**

   From the terminal, run:

```
spark-submit gradient_boosting_model.py
```

5. **Monitor the output**

   Your script will print outputs to the terminal.

   If errors occur, check:

   - File paths in HDFS
   - SparkContext availability.

Please note that the python codes for the regression models have been uploaded to the GitHub site - **https://github.com/Soligrl/CIS-5560**

References

Data Source:
Redfin Housing Market Data – Kaggle

GitHub Repository (Project Code):
https://github.com/Soligrl/CIS-5560

Professor's Lab Tutorial and Course Lectures:
CIS 5560 – Introduction to Big Data Science, Spring 2025.
Lab materials and lecture notes provided by Prof. **Jongwook,Woo, PhD**, California State University, Los Angeles.