Big Data HW6

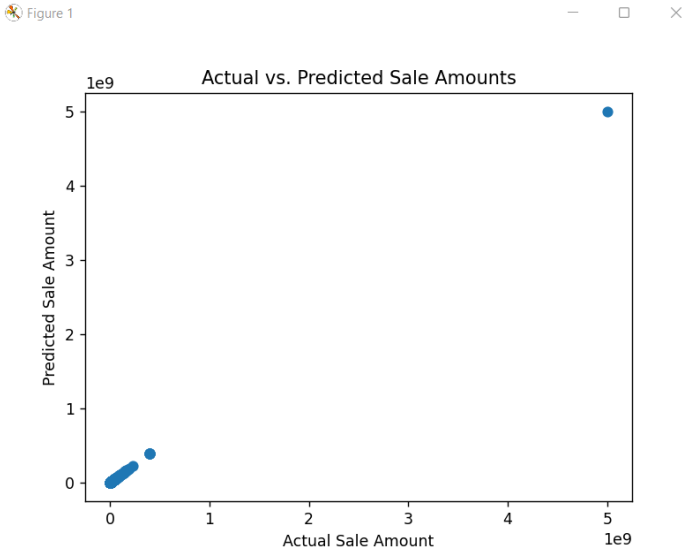
Nektario Hajiyianni U221N0154

Option 2: Big Data Processing using Spark

Code

All code is available on GitHub: https://github.com/tario-hajiyianni/Real_Estate_HW6

Spark:



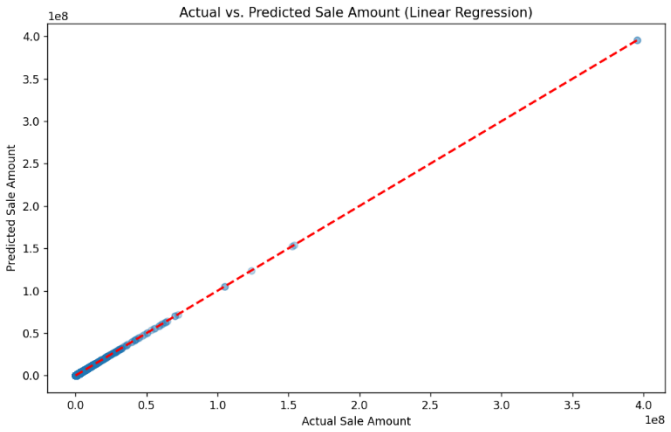

```
Root Mean Squared Error (RMSE): 0.10044975601020611
24/05/12 14:59:49 WARN GarbageCollectionMetrics: To enable non-built-in garbage collector(s) List(G1 Concurrent GC), users should configure it(them) to spark.eventLog.
gcMetrics.youngGenerationGarbageCollectors or spark.eventLog.gcMetrics.oldGenerationGarbageCollectors
Execution Time: 28.245131969451904 seconds
SUCCESS: The process with PID 45984 (child process of PID 45140) has been terminated.
SUCCESS: The process with PID 45140 (child process of PID 30316) has been terminated.
SUCCESS: The process with PID 30316 (child process of PID 26764) has been terminated.
```

Spark Implementation Output:

Root Mean Squared Error (RMSE): 0.10044975601020611

Execution Time: 28.245131969451904 seconds

Single Threaded:




```
Train MSE: 1.1997882037265381e-15
Test MSE: 9.046120363762368e-17
Train R^2 Score: 1.0
Test R^2 Score: 1.0
Execution Time: 1.944549322128296 seconds
```

Single-Threaded Application Output:

Train MSE: 1.1997882037265381e-15

Test MSE: 9.046120363762368e-17

Train R^2 Score: 1.0

Test R^2 Score: 1.0

Execution Time: 1.944549322128296 seconds

Comparison:

Performance: The single-threaded application outperforms the Spark implementation in terms of execution time, taking only 1.944 seconds compared to Spark's 28.245 seconds. This indicates that for this specific task and dataset size, a single-threaded approach is much faster.

Resource Utilization: The Spark implementation likely utilizes more computational resources due to its distributed nature, which may explain the longer execution time compared to the single-threaded application. However, it also offers parallel processing capabilities, enabling it to handle larger datasets efficiently.

I believe that Spark would be faster then the Single-Threaded version of it had more sufficient resources than just my machine like a machine that would has better resources or a distributed network of machines.

Accuracy: Both implementations achieve excellent accuracy, as indicated by the very low MSE and perfect R^2 scores in the single-threaded application. The Spark implementation also achieves a relatively low RMSE, suggesting good predictive performance.

Scalability: While the single-threaded approach is faster for this dataset size, Spark's strength lies in its scalability. For larger datasets or more complex computations, Spark's distributed computing capabilities can provide better performance than a single-threaded approach.

Additional Notes:

In Dataset_Preprocessing.py , I split the data into one dataset with locations (co-ordinates) and one without. I was initially planning to use the location dataset to try do location mappings and analysis like these examples :
https://ncar.github.io/PySpark4Climate/tutorials/pyspark-geo-analysis/geopandas-and-spark/

https://medium.com/ibm-data-ai/analyzing-geospatial-data-in-apache-spark-f638601e405a

I was planning on attempting to try the following options:

- Patterns of property development and urbanization by analyzing the distribution of assessed values and sale amounts across different towns and residential areas.
- Spatial clustering to identify areas of high and low property values
- Analyze property sales ratios across different towns and property types to identify trends in real estate markets and assess market competitiveness.

I couldn't get it working in time.
However, I may attempt to add it to my Final Year Project as this dataset is from my project.