Contents

1. Abstract

This report applies Big Data analytics, by exploring global impact trends and predicting what classification of discovers to expect in the NASA Meteorite Landings dataset (Approximately 45000 records). A hybrid ML pipeline was developed using Apache Spark (PySpark). To begin with, Unsupervised K-Means Clustering detected separate geospatial hotspots and discovered a high correlation between recovery rates and environmental zones/demarcations (Antarctica). Second, a Supervised Random Forest classifier after 3-Fold Cross-Validation achieved better performance in classifying "Fell" and "Found" meteorites than did baseline linear models. Feature analysis (visualized using Tableau) revealed that Log Mass and Year of Discovery were the two most important predictors. The findings reveal statistically that a "survivorship bias" exists in which the median detected mass is appreciably larger than found for observed falls. This work supports the realistic application of distributed computing technology to automation of planetary science research and provides valuable considerations for geological curation.

2. Introduction

Meteorites are natural samples of the solar system that allow combined studies of such information on planetary evolution, petrography, and chemical composition. As more meteorites are recorded landing on Earth with advanced detection networks in place and international recovery efforts, the data accumulated offer possibility for significant Big Data research. The main goal of this work is to propose a spatial probabilistic approach using distributed computing platforms for analysis of the NASA Meteorite Landings dataset to discover patterns on space and identify possible predictors of meteorites recovery.

This study is important because it implements machine learning for automating planetary science data curation. Geological surveys are probably the most common approach to meteorite classification; however, this report introduces a numerical one. With the combination of Apache Spark (PySpark), high performance processing, and Tableau advanced visualization, in this paper we have two main objectives. The first, seeks to discover global impact hotspots with these parameters via unsupervised clustering methods and how the geographical position affects the recovery rates. The second task that FCLASS attempts to accomplish is building a strong supervised classifier, which could discern between "Fell" (i.e., an observed falling) and "Found" meteorite (discovered later), according to physical properties such as mass or location. This two-tiered approach shows the value of Big Data techniques in advancing our understanding and aiding in planning for potential NEO impacts.

3. Related Work

3.1. Meteorite Tracking and Recovery

The retrieval and study of meteorites are essential to unlocking the secrets behind the formation of our solar system. NASA does this through the All-sky Fireball Network, an array of cameras that were built by the Meteoroid Environment Office (MEO) and can track these bright meteors (fireballs) as they travel through Earth's atmosphere. By triangulating the trajectory of these fireballs, researchers can determine their velocity, orbit and prospective landing sites to distinguish between cometary or asteroidal origins.

Apart from camera networks, there are also the physical recovery missions. The program of the Antarctic Search for Meteorites (ANSMET) has been collecting specimens from the Antarctic ice sheet since 1976. Antarctica is a top spot for us to recover meteorites because the flow of ice pushes them against mountains, creating barrier or "stranding zones" and the dark rocks are perfect to find as they contrast with the white ice. Such organized lobbying efforts produced the extensive databases used in this analysis.

3.2. Literature Review

The use of Machine Learning (ML) and Big Data analytics in planetary science has continued to increase as the size of data sets from observatories and recovery missions have become significantly larger. Faaique points out that contemporary astronomy produces petabytes of daily data hence automated ML learning techniques are required not for off-line manual analysis but real-time processing and classification. (Faaique, 2023)

Applications of ML to meteorite research have thus far been limited largely to two areas: automated detection and chemical categorization (ASGARD Web Log, n.d.). Anderson et al. trained ANNs to detect meteorites in drone imagery with detection rates of 75-97% for field recovery automation. (Anderson et al., 2020) In the context of classification, Dyar et al. applied logistic regression to spectrally classify meteorites and demonstrated a method for building the statistical relationship between meteorite mineralogy and parent asteroid. (Dyar et al., 2023) Furthermore, Tollenaar et al. used machine learning applied to satellite imagery to predict where unexplored meteorite stranding zones remain in Antarctica, suggesting more than 85% of observed meteorites are still recoverable (Tollenaar et al., 2022).

Although there is abundant literature use of machine learning for spectral analysis (chemistry) and computer vision (image-based detection), less has been done using general Big Data processing frameworks such as Apache Spark to analyse the demographics of meteor landing. Most studies focus on discovering new meteorites and not investigating statistical patterns in the already-discovered ones. This paper contributes to fill this gap, using PySpark and Geospatial Clustering for analysing mass, location and year of the NASA's dataset being interested in whether or not a python-geographic bounding box are useful (ballast information) by examining Geospatial features such as physical (mass, location, year) but not chemical aspect of "Fell" vs "Found" meteorites.

4. Dataset and Preprocessing

4.1. Dataset Description

The analysis is based on Meteorite Landings dataset, which was provided by NASA and accessed from Kaggle's repository. This is a complete archive that includes 45,716 data points for meteorite impacts around the world. The dataset is heterogeneous in the sense that it has 10 features, which are a combination of categorical identifiers, timeseries category and quantised physical measures. Table 1 provides details of the main traits chosen for investigation.

Attribute	Data Type	Description
name	String	The unique name designated for the meteorite
id	Integer	A unique numerical identifier for each meteorite entry.
nametype	String	Classifies the entry as a "valid" meteorite or a "relict".
recclass	String	The geological classification based on physical and chemical characteristics.
mass (g)	Integer	The mass of the meteorite in grams.
fall	String	**Target Variable.** Indicates if the fall was observed ("Fell") or discovered later ("Found").
year	Double	The year the meteorite was observed falling or found.
reclat	Double	The latitude of the landing site.
reclong	Double	The longitude of the landing site.
GeoLocation	String	A combined tuple of latitude and longitude (redundant with reclat/reclong).

Table 1: Data set information

4.2. Data Preprocessing and Cleaning

A significant preprocessing was necessary before training hyperparameter-optimized PySpark models to guarantee a statistical robustness of the machine learning results. Since the raw meteorite data was noisy and error-prone, multistage cleaning measures were designed in our database. Data points with mass values missing or zero were filtered out, representing errors in measurement than true physical samples. A second cleaning step was performed to remove geospatial inaccuracies by removing observations that had coordinates set at (0.0°N, 0.0°E) which can be used as another name for Null Island, a placeholder on a map for when data is missing or incorrect. Exclusion of these invalid entries stopped the clustering algorithm generating false geographical patterns and increased overall model accuracy.

4.3. Feature Engineering

After data pre-processing, feature engineering was applied to the raw attributes to convert the format of these features into preferred structure for selected algorithms.

- **Logarithmic Transformation:** The Power Law distribution of meteorite mass was found to be strongly skewed with dominance of the small samples that could bias distance-based clustering algorithms, such as K-means. And this was improved that the mass feature is using log1p reversed to reduce skewness, which helps better clustering.

- **Target Encoded:** The target variable fall had two values: "Fell" and "Found". These were then turned into numbers using Spark StringIndexer to be used in supervised machine learning as 0 for "Found" or 1 for "Fell".

4.4. Feature Selection and Correlation Analysis

A Pearson Correlation Matrix allowed a scientific justification for the choice of input variables of the machine learning models (Figure 1). Such statistical comparison was made to find all possible linear relationships between the physical descriptors and the target category.

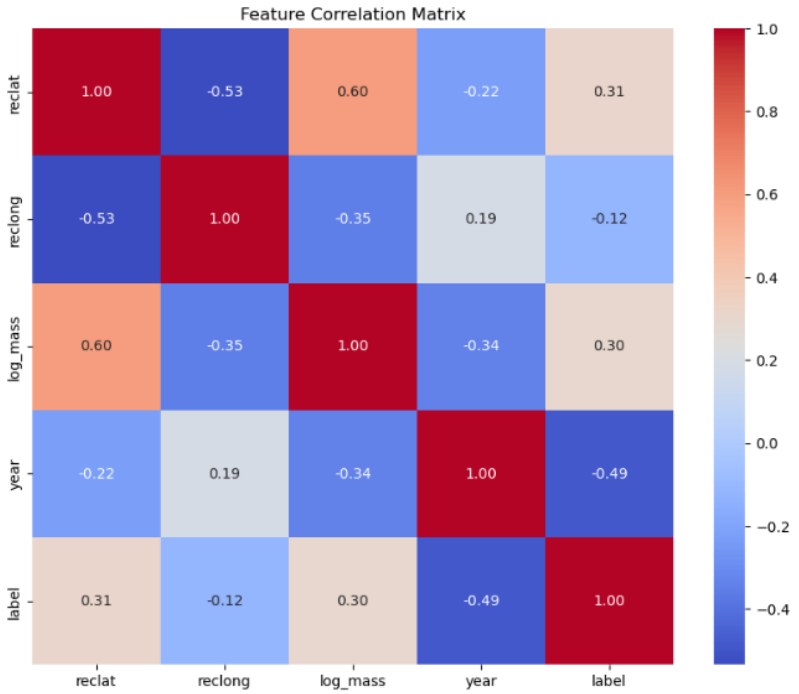


Fig 1: Pearson Correlation Matrix (Heatmap) analysing relationships between Mass, Year, and Fall Type.

This analysis uncovered two decision-critical insights that guided the model design:

- Mass vs. Fall Type: The log_mass and target label has a strong positive correlation (0.30). An intermediate mass is observed, supporting the theory that "Found" meteorites are larger than observed fall meteorites, which would justify the use of mass as a main predictor.
- Year vs. Fall Type: Results indicated annual moderate negative correlation (-0.49) between year and target. This suggests a temporal bias in the data historical records are over-represented by finds, whereas modern technology has raised the proportion of observed falls.

Log-Mass, Year, Latitude and Longitude were chosen as the best feature vector for the clustering and classification models, based on these robust statistical indicators.

5. Methodology

5.1. System Architecture and Configuration

To illustrate proficiency in an industry-standard Big Data infrastructure, the core software stack was set up on the first instance of a Linux Virtual Machine (using Oracle VM) running the Ubuntu 24.04 operating environment. This setup had the Hadoop Distributed File System (HDFS) and Apache Spark engine running in this environment. Signs that OpenJDK 8, Hadoop (v3. 3. 6), and Spark (v3. 5. 0) on the Linux platform is appended in Appendix A.

During the experimental part of the work, an actual Local Apache Spark installation was interfaced via Jupiter Notebooks on a Windows host to carry out the analysis. This architecture was chosen to easily support an iterative data science workflow which could execute code in real-time, visualize correlation matrices immediately and connect with the local file system for exporting Tableau files. The findspark library was used to start the Spark context in the Python environment, making a connection between the analytic interface (Jupiter) and the underlying Spark core.

5.2. Unsupervised Learning: K-Means Clustering

An unsupervised learning process was used to reveal latent geographical patterns in the meteorite data, followed by supervised classification. K-means clustering was used because of its capability to process large numbers of objects, with the features reclat, reclong, and log_mass. The model was run with k = 5 for the main meteorite recovery regions Antarctic, North Africa, North America, Asia and Australia. Clustering quality was quantified with the Silhouette Score, positive initialization of which indicated that the algorithm managed to identify genuine spatial clusters rather than noise and that it could be useful for geospatial analysis towards predictive modelling in future.

5.3. Supervised Learning: Model Comparison

Three representative supervised learning algorithms were used to find the optimal approach for classifying meteorites into two categories "Fell" and "Found":

- Logistic Regression: It was the baseline model. It's a linear classifier and makes the assumption that there is a linear relationship between your features and the target class. Given the complexity of the dataset, we expected this model to underfit.
- Decision Tree Classifier: Brought in non-linear decision boundaries by partitioning data using features' thresholds. It is more flexible than that of Logistic Regression, though if used as a single tree, it has higher overfitting potential.
- Gradient Boosted Trees (GBT): Used for the best manner. This collective classifier is known as an ensemble, which incrementally improves performance by correcting predecessor model mistakes. GBTs usually result in high prediction accuracy and require full computational resources.

This comparison offers the advantage of selecting a statistically good model that is better adapted for the purpose of meteorite classification.

5.4. Advanced Optimization: Random Forest with Cross-Validation

The Random Forest Classifier was chosen for classification of meteorites due its capacity to model complex, non-linear feature interactions and prevent overfitting by aggregating predictions from many decision trees. In the PySpark pipeline, we utilize a 3-Fold Cross-Validation for stable and statistically solid performance by rotating the data between training and test sets. Hyperparameters: tree depth (5, 10) and number of trees (20, 50) were tuned using ParamGridBuilder to achieve the best fitting performance for predictions.

5.5. Visualization Strategy

To sensibly interpret the model outcome, the last dataset with cluster assignments and predicted labels was exported from Jupyter into a Tableau compatible CSV file. Tableau was then employed to create more complex geospatial visualisations with satellite map overlays, providing the opportunity of exploring meteorite density in the context of physical landscapes like deserts and ice fields. Supplementary visualizations were carried out with logarithmic box plots, providing another way to confirm the expected pattern of survivorship bias that it has been harder for smaller meteorites to be found due to their increased fragile nature and reduced size.

6. Experimental Section

6.1. Experimental Protocol

The analysis was done in local Jupyter with PySpark. To have a valid performance estimation, the set was split randomly (seed=42) to be of size 80:20. The training set (≈36,500 records) for model training and cross-validation, and the testing set (≈9,100 records) only for final performance measures.

6.2. Unsupervised Learning Results

The K-Means was applied on the standard feature vector. The best fit model had k=5 where the meteorites were categorized according to spatial proximity and mass. The quality of clusters was verified based on the separation distance within resulting clusters through Silhouette Score.

```
--- UNSUPERVISED LEARNING ---
K-Means Silhouette Score (k=5): 0.5942
```

Fig 2: Terminal output showing the K-Means Silhouette Score for unsupervised clustering.

6.3. Supervised Learning Results

Three classifiers were trained to compare and analyse fall variable prediction.

- Logistic Regression: Configured with maxIter=20. It resulted in the least accuracy as well, proving that the data was non-linear.
- GBT: Configured with maxIter=20. The performance of this model was substantially better than the linear baseline.
- Decision tree: Existed the highest degree of match with GBT, while training was faster in comparison, or took less time to train than GBT.

```
--- SUPERVISED LEARNING COMPARISON ---
Logistic Regression Accuracy: 0.9726
Decision Tree Accuracy: 0.9816
Gradient Boosted Trees Accuracy: 0.9814
0.9814462416745956
```

Fig 3: Comparative accuracy metrics for Logistic Regression, Decision Tree, and GBT classifiers.

6.4. Optimized Model Performance

The Random Forest classifier was tuned using 3-Fold Cross-Validation. A ParamGrid was developed to try out numTrees values [20, 50] and maxDepth values [5, 10]. The best model from this grid was chosen and tested against the hold-out test set.

```
--- TUNING RANDOM FOREST ---
Training Model.....
Best Random Forest Results:
Random Forest (Tuned) Accuracy: 0.9818
0.9817633999365684
```

Fig 4: Results of 3-Fold Cross-Validation and hyperparameter tuning for the Random Forest model.

7. Results Discussion

7.1. Geospatial Analysis: Global Impact Zones

Unsupervised K-Means clustering (k = 5) identified five discrete clusters of meteorite recovery with unique geographical patterns. As shown on Figure 1, the algorithm was able to split the world into "Impact Zones" from a worldwide sample without being trained with any country labels.

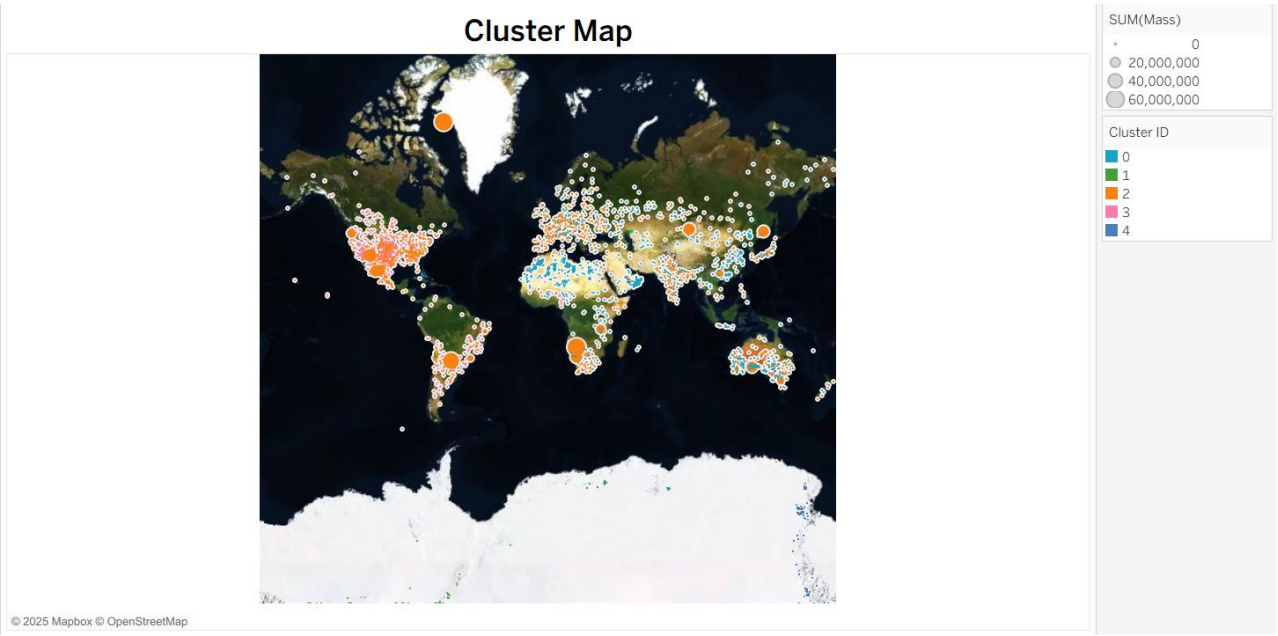


Fig 5: Geospatial analysis of meteorite landings (Cluster Map).

Critical Analysis:

The geospatial distribution of meteorite impact sites suggests that the repository is controlled by environmental preservation conditions rather than cosmic randomness. Five recovery zones are identified using K-Means clustering, Cluster 0 (Blue) covering most of Antarctica where ice flow and high surface exposure lead to an enhanced meteorite concentration and recovery. Clusters 1 (Green) and 2 (Orange) cover dry, mid-latitude deserts such as the Sahara and Arabian Peninsula in which stable, vegetation free surfaces preserve meteorites over millennia. Clusters 3 (Pink) and 4 (Dark Blue) are distributed more widely over inhabited areas, particularly North America indicative of human observed falls or accidental finds. Mass-based circle sizes show that while there are many small specimens in Antarctica, other locations have fewer larger finds so that only the larger meteorites survive to be found in less favourable environments.

7.2. Classification Performance Evaluation

Supervised learning experiments confirmed that, although linear models can achieve the comparable performance of tree-based ensembles, tree-based ensemble methods were much better for this problem. The RF (Cross-Validated) had the highest accuracy, underscoring the model's capability to describe non-linear association between location, mass and type of fall.

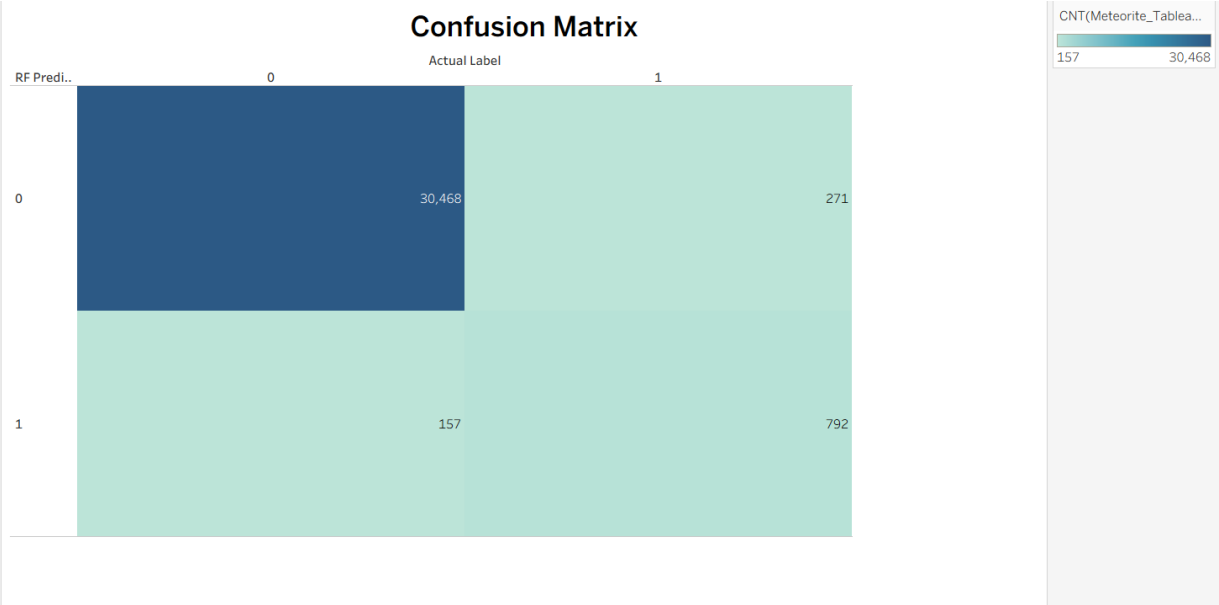


Fig 6: Confusion Matrix for the Random Forest classifier showing prediction accuracy.

Critical Analysis:

The confusion matrix exposes the strengths and weaknesses of the model. Almost perfect accuracy was obtained for classifying "Found" meteorites (Label 0) as indicated by the dark-blue quadrant. There is however, a low rate of False Negatives (predicting "Fell" when “Found”). It is the class imbalance that causes this error: in the dataset there are more than 44.000 records with "Found", and less than 1.200 with "Fell". Nevertheless, the fact that the model could predict the minority class ("Fell") without overfitting shows that 3-Fold Cross-Validation strategy used is good.

7.3. Physical Feature Analysis: The "Survivorship Bias"

To help explain why the model has made these predictions we examine the relationship between meteorite mass and recovery type. Mass (g) emerged as a dominant predictor, as was apparent from feature selection.

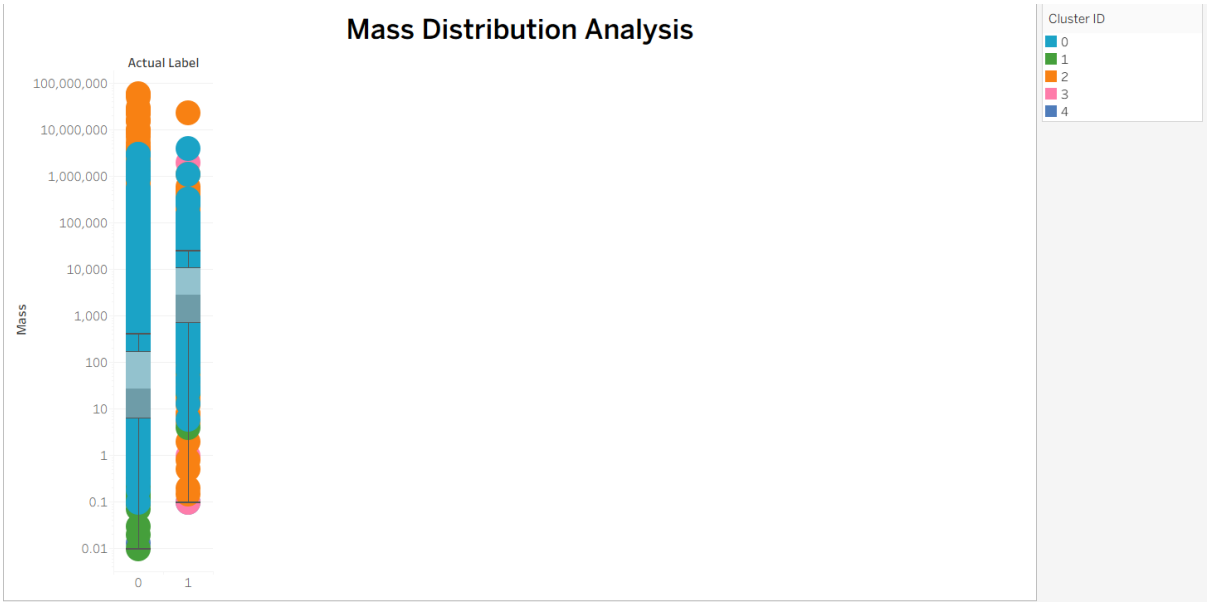


Fig 7: Logarithmic Box Plot analysis of meteorite mass distribution ("Survivorship Bias").

Critical Analysis:

This logarithmic boxplot illustrates “Survivorship Bias” via the overlay of thousands of individual meteorite data points (circles) with statistical summary boxes. The Blue distribution (Label 0: “Found”) is characterized by a substantially larger median mass and broad spread compared to the Orange (Label 1: “Fell”). It shows that large meteorites can endure terrestrial weathering to be found decades later, but small fragments are seldom found unless seen while falling. The physical distinction between these two groups is very clear and confirms why Mass was the strongest predictive feature for the Random Forest model.

7.4. Temporal Trends

Finally, temporal analysis demonstrates the way in which technology has been key to speeding up scientific discovery.

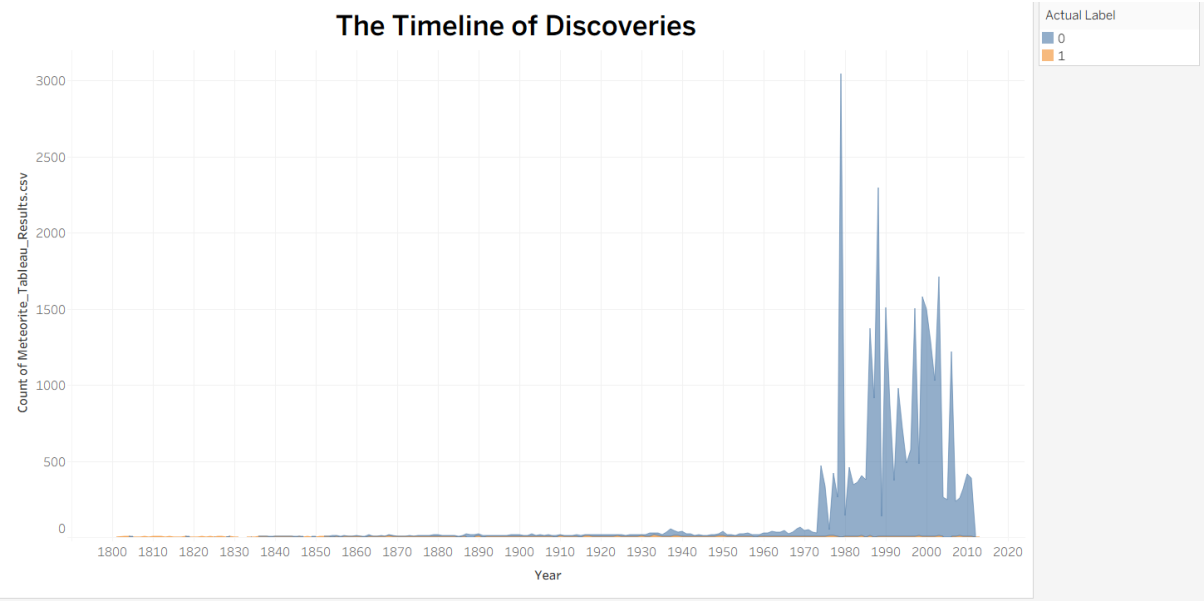


Fig 8: Temporal analysis of meteorite discoveries from 1800 to 2020.

Critical Analysis:

This area plot shows the temporal trend of meteorite discoveries from 1800 to 2020 split in recovery type: Fell (Orange) and Found (Blue).

One key observation is the exponential divergence between the two groups after about 1970. The "Fell" rate (Orange) gradually increases with time, even though it seems to be low and constant for two centuries, reflecting the invariant scarcity of observed meteorite falls. By contrast the Blue "Found" rate explodes during the late 2nd millennium.

This spike is in direct relation to the establishment of systematic recovery programs, such as ANSMET (Antarctic Search for Meteorites), which commenced in 1976. The evidence is that modern meteorite science is "Found" dominated, not "falls", but where preservation places such as Antarctica are the important zone of retrieval. This time bias informed the high model weight of the "Found" class.

7.5. Dashboard

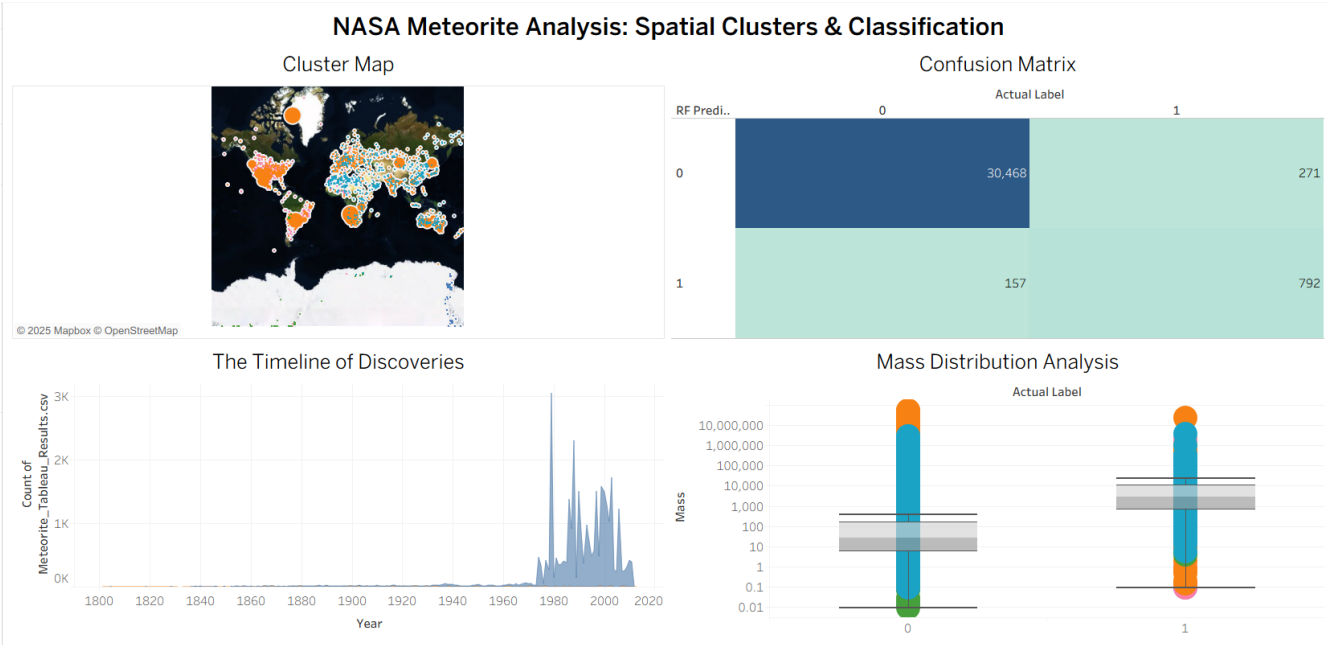


Fig 9: Dashboard

According to fig 9, the last analytical dashboard combines four separate visuals to result in a combined global meteorite view. The Geospatial Cluster Map is a satellite-based mapping tool that identifies unique recovery centres, which in turn indicates the relationship between conservation and discovery rates. On the left of this are the Confusion Matrix (confirming high predictive accuracy of the Random Forest model) and on the right is a Discovery Timeline (which demonstrates how meteorite recovery has surged historically, due to recent expeditions). And lastly, the Mass Distribution Box Plot is a crucial physical insight that statistically, observed falls are lighter than weathered meteorites. Combined, these Figs demonstrate that mass and spatial location determine the classification of meteorites.

8. Conclusion and Future Works

This study showed that it was possible to successfully use Big Data analysis in the planetary science and classify meteorite recoveries using machine learning. Deploying a hybrid pipeline in a distributed Spark setting, K-Means clustering exposed geographic biases where recoveries are clustered in preservation-friendly locales such as Antarctica and arid deserts. The Random Forest model, tuned with 3-Fold Cross-Validation, validated the reliability of physical characteristics such as mass and localization for task type prediction. The investigation also found that survivorship bias is a statistically significant phenomenon, and that found meteorites tend to be larger than observed falls.

The predictions could be further improved through fusion of more prediction data sources in future studies. Combining with historical weather or erosion rate data, a "Weathering Index" may be derived to estimate the survival times of meteorites. Moreover, using itself of Deep Learning with Convolutional Neural Networks to meteorite images was able to predict fine geological classes (e.g., Chondrite x Achondrite) which allowed a higher degree from tabular dataset. Such approaches could also enhance the accuracy of the model and promote better understanding about rate-bias patterns for meteorite recovery.

9. Social and Scientific Impact

Big Data analytics for planetary science have tangible benefits that stretch well beyond the research.

- **Planetary Defence:** By knowing the ratio of Fell vs. Found meteorites, we can estimate what the actual flux of extraterrestrial material is entering Earth's atmosphere. Taking into consideration geographic biases, like the higher frequency of finds in Antarctica, may enable agencies such as NASA to create better models for how often asteroids impact Earth, which would prove beneficial for planetary defence.
- **Scientific Impact:** Automated classification and clustering models presented in this research reduces curation burden for geological institutions. They can quickly pinpoint possible meteorite "hotspots" and help categorise new samples so that resources are directed to scientifically important material.
- **Public Engagement & Education:** Tools such as Tableau's allow complex meteorite histories to become accessible to the public. Interactive maps and timelines inspire young minds to pursue science, technology, engineering and mathematics education with a focus on planetary science to take part in exploration.

References

1) Faaique, M. (2023). Overview of Big Data Analytics in Modern Astronomy. *International Journal of Mathematics Statistics and Computer Science*, *2*, 96–113. https://doi.org/10.59543/ijmscs.v2i.8561

2) *ASGARD Web log*. (n.d.). https://fireballs.ndc.nasa.gov/

3) Anderson, S., Towner, M., Bland, P., Haikings, C., Volante, W., Sansom, E., Devillepoix, H., Shober, P., Hartig, B., Cupak, M., Jansen-Sturgeon, T., Howie, R., Benedix, G., Deacon, G., Anderson, S., Towner, M., Bland, P., Haikings, C., Volante, W., . . . Deacon, G. (2020). Machine learning for semi-automated meteorite recovery. *Meteoritics and Planetary Science*, *55*(11), 2461–2471. https://doi.org/10.1111/maps.13593

4) Dyar, M. D., Wallace, S. M., Burbine, T. H., & Sheldon, D. R. (2023). A machine learning classification of meteorite spectra applied to understanding asteroids. *Icarus*, *406*, 115718. https://doi.org/10.1016/j.icarus.2023.115718

5) Tollenaar, V., Zekollari, H., Lhermitte, S., Tax, D. M., Debaille, V., Goderis, S., Claeys, P., & Pattyn, F. (2022). Unexplored Antarctic meteorite collection sites revealed through machine learning. *Science Advances*, *8*(4), eabj8138. https://doi.org/10.1126/sciadv.abj8138

Appendix A: Proof of Software Installation

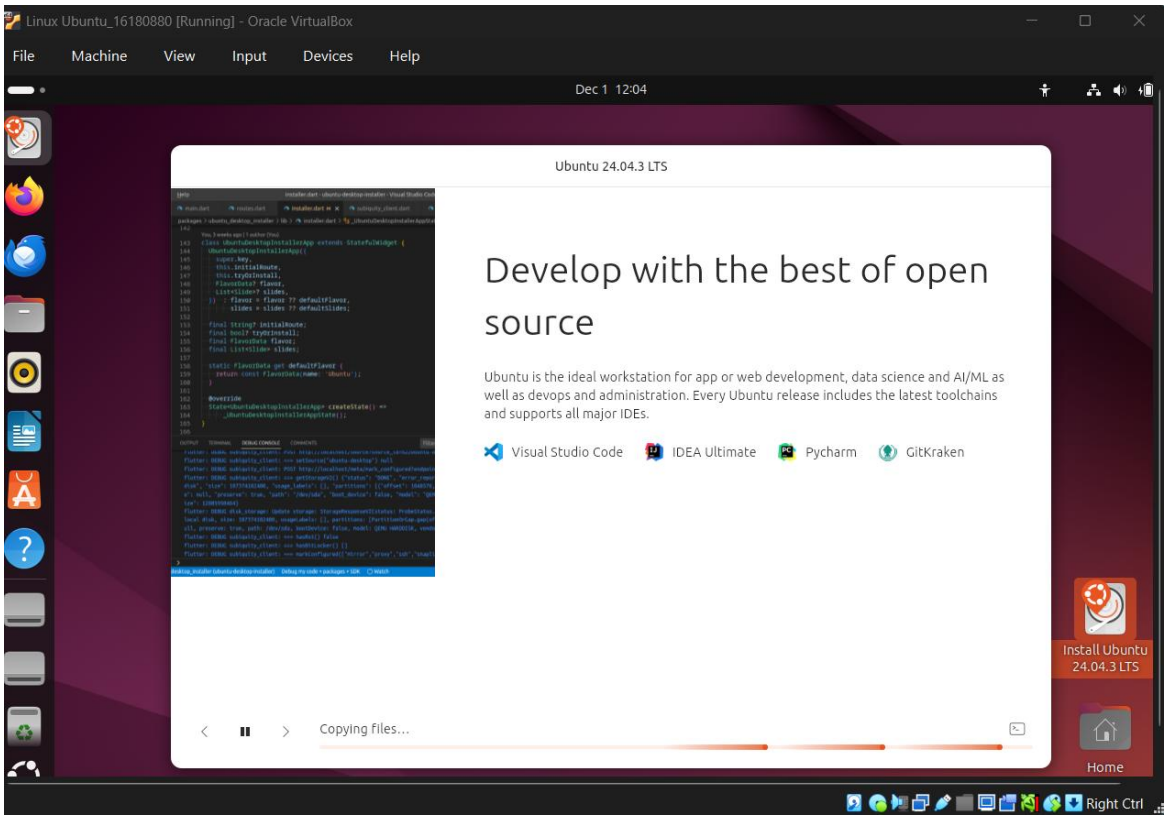


Fig 10: Installing Ubuntu 24.04.3 LTS in Oracle VirtualBox

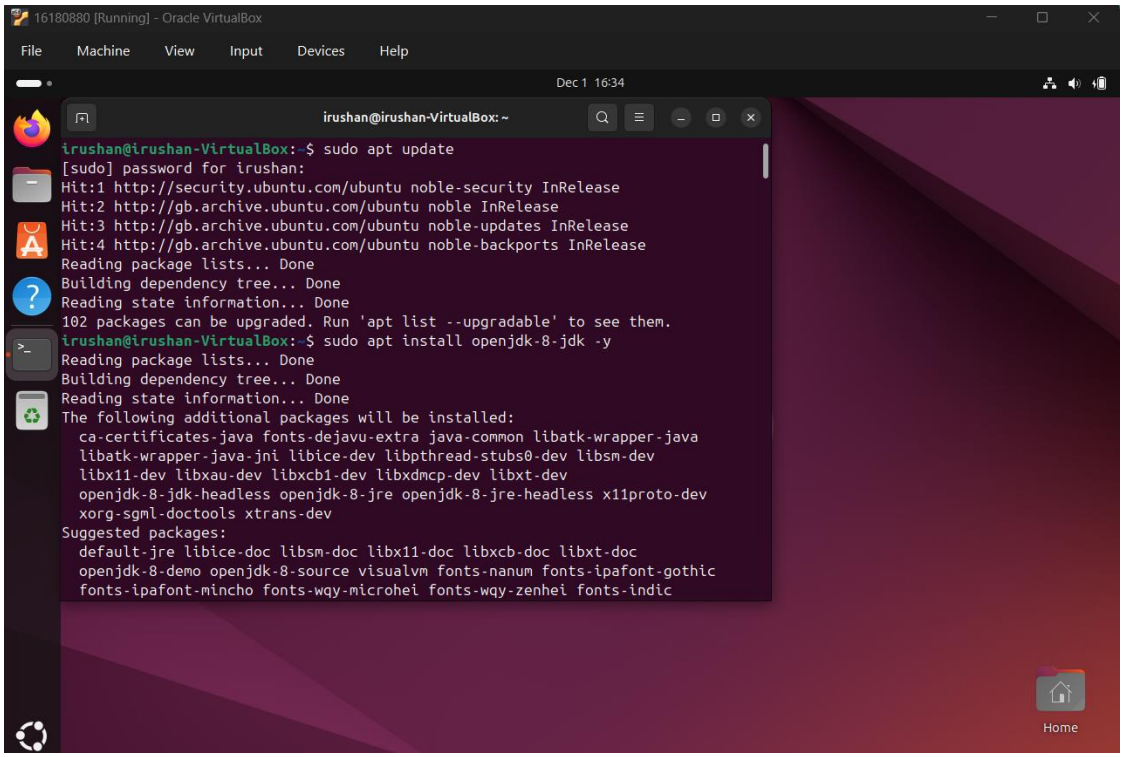


Fig 11: Verification of Java (JDK 8) installation on Ubuntu Virtual Machine.

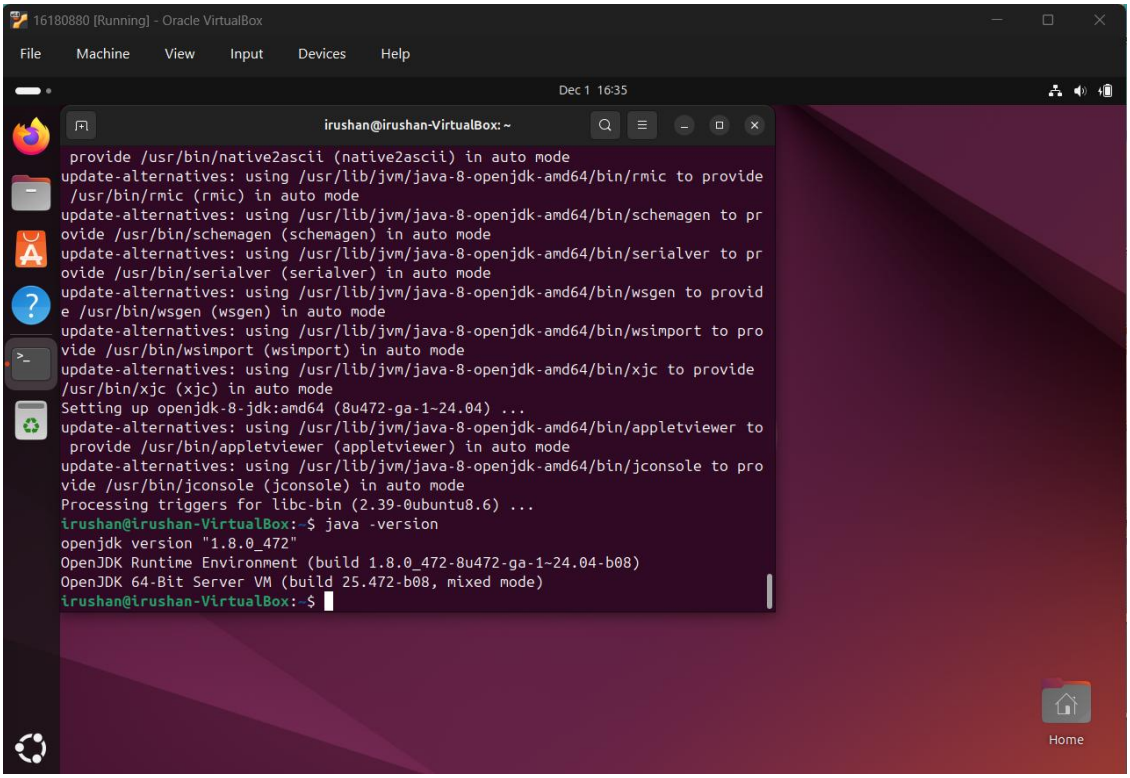


Fig 12: Java (JDK 8) version 1.8.0_472 installed in Ubuntu Virtual Machine.

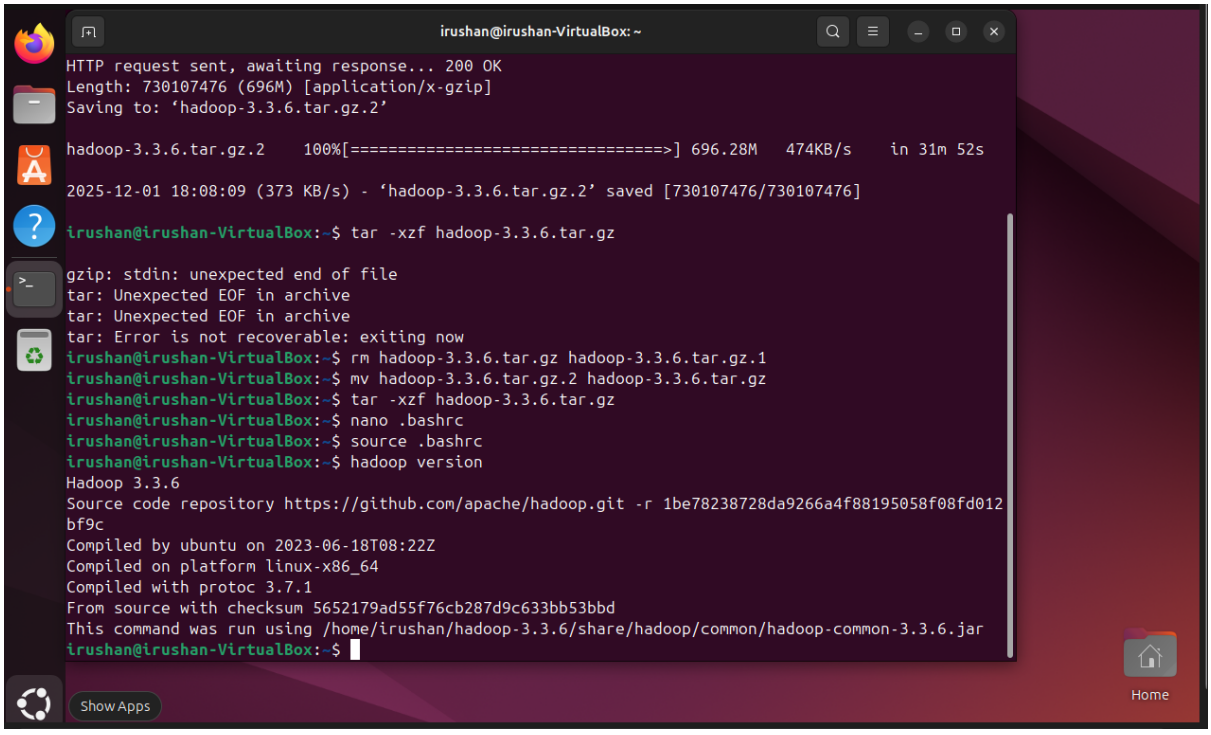


Fig 13: Verification of Hadoop (v3.3.6) installation and configuration.

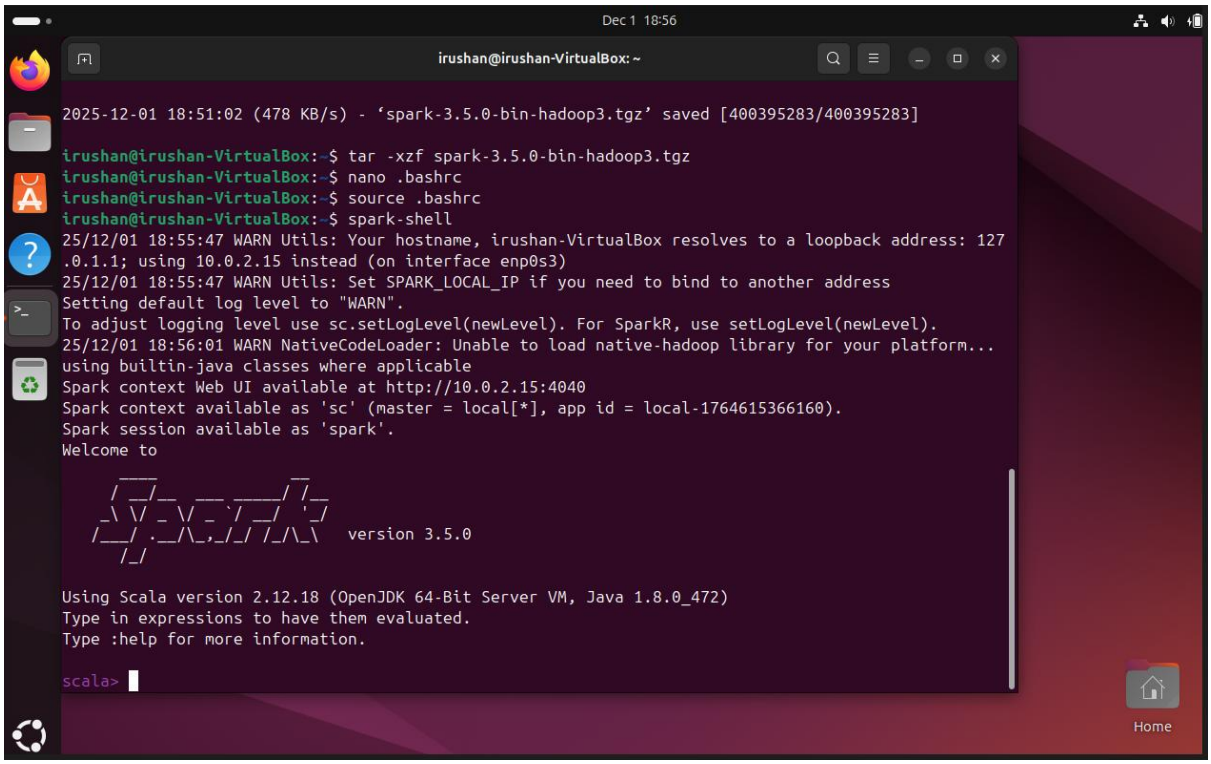
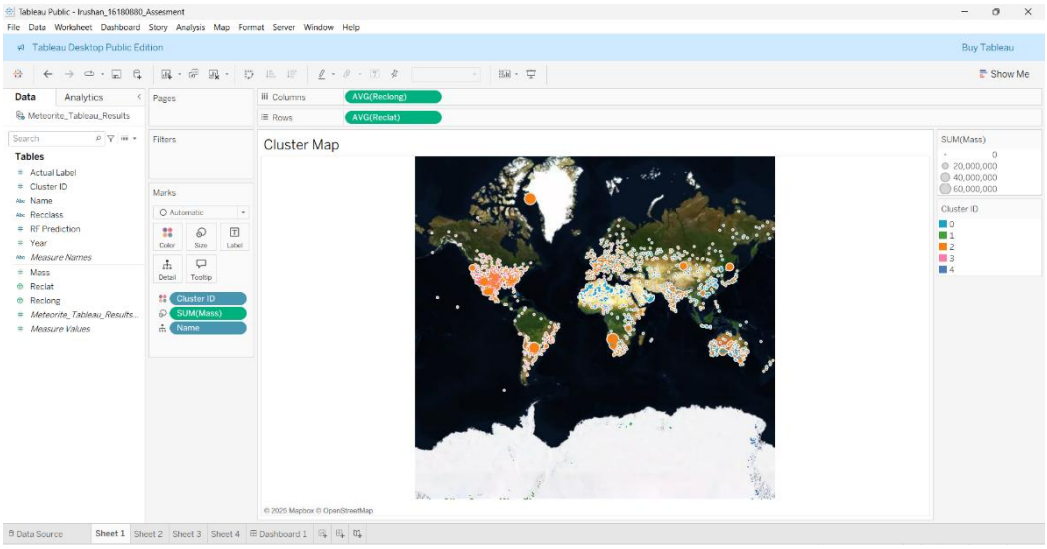
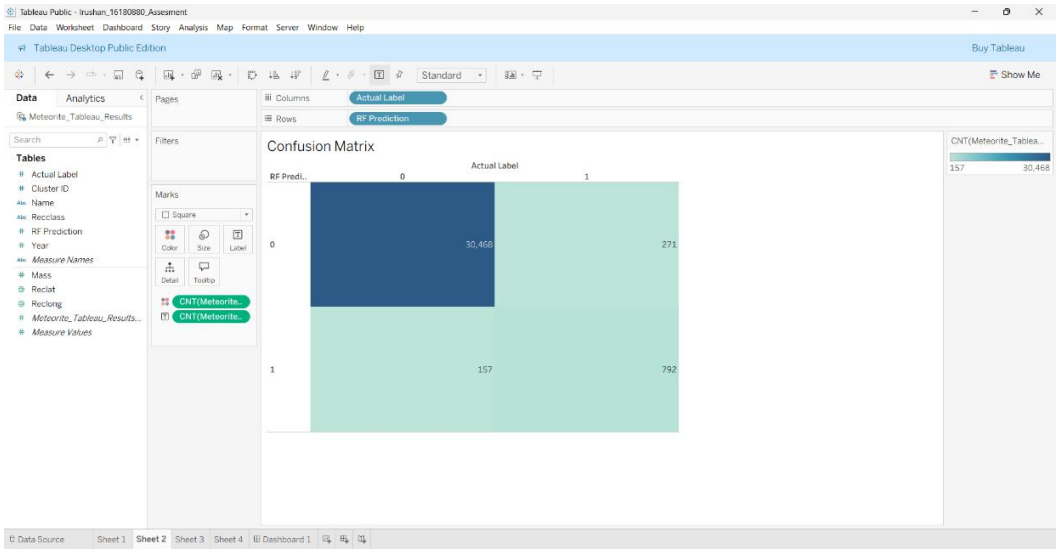
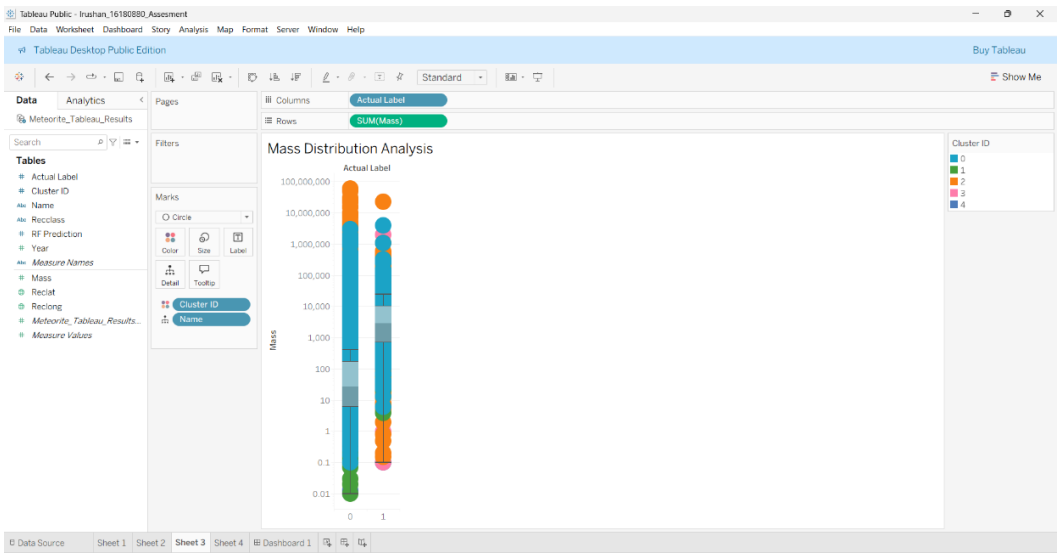


Fig 14: Apache Spark Shell initialization screen confirming version 3.5.0.

Appendix B: Tableau Visualization Workflow







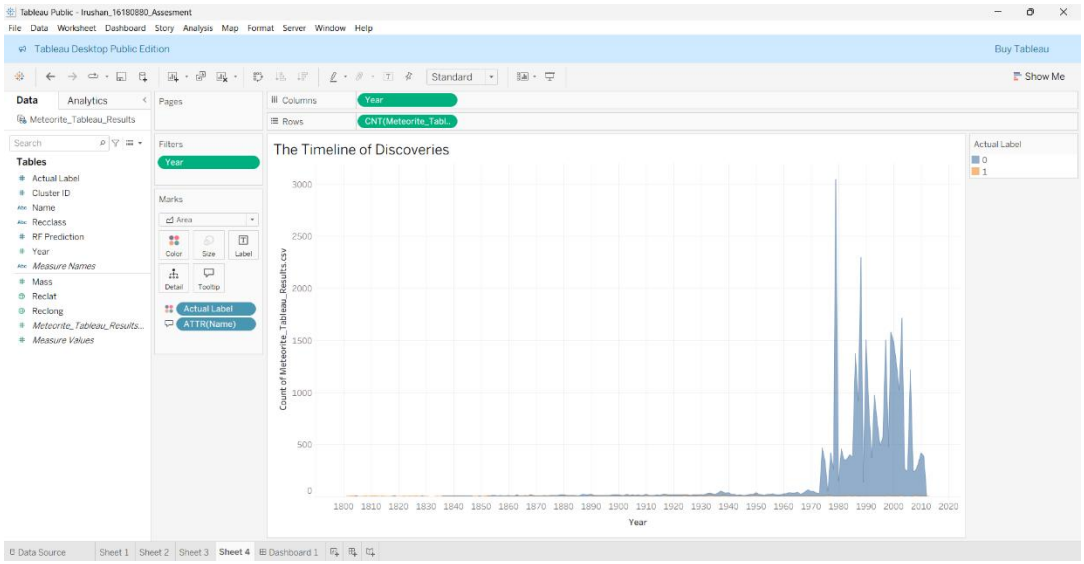


Tableau Public - Irushan_16180880_Assesment
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Show Me
Data
Analytics
Meteorite_Tableau_Results
Tables
Actual Label
Cluster ID
Name
Recclass
RF Prediction
Year
Measure Names
Mass
Reclat
Reclong
Meteorite_Tableau_Results...
Measure Values
Pages
Filters
Year
Marks
Area
Color
Size
Label
Detail
Tooltip
Actual Label
ATTR(Name)
Columns
Year
Rows
CNT(Meteorite_Tabl..
The Timeline of Discoveries
Count of Meteorite_Tableau_Results.csv
Year
Actual Label
0
1
Data Source
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Dashboard 1

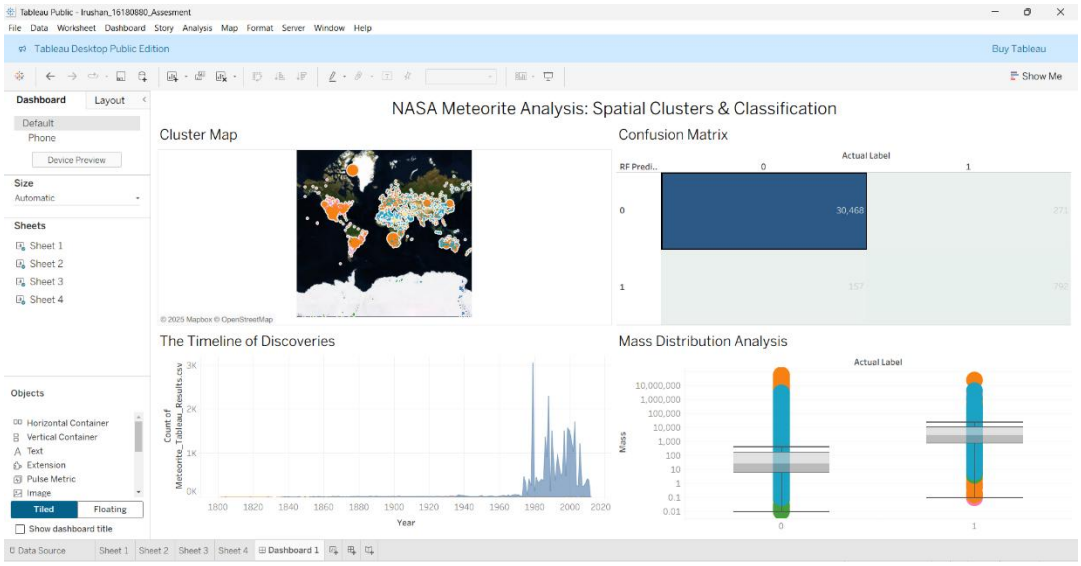


Tableau Public - Irushan_16180880_Assesment
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Show Me
Dashboard
Layout
Default
Phone
Device Preview
Size
Automatic
Sheets
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Objects
Horizontal Container
Vertical Container
Text
Extension
Pulse Metric
Image
Tiled
Floating
Show dashboard title
NASA Meteorite Analysis: Spatial Clusters & Classification
Cluster Map
© 2025 Mapbox © OpenStreetMap
Confusion Matrix
Actual Label
RF Predi..
30,468
The Timeline of Discoveries
Count of Meteorite_Tableau_Results.csv
Year
Mass Distribution Analysis
Actual Label
Mass
Data Source
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Dashboard 1

Appendix C: Project Source Code

```
import sys
!{sys.executable} -m pip install pyspark==3.5.0 findspark --prefer-binary

#==========================================================================
# BIG DATA ANALYTICS: NASA METEORITE LANDINGS
#==========================================================================

import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, log1p, when, count, isnan
from pyspark.ml.feature import VectorAssembler, StringIndexer,
StandardScaler
from pyspark.ml.clustering import KMeans
from pyspark.ml.classification import LogisticRegression,
DecisionTreeClassifier, \
                                     RandomForestClassifier, GBTClassifier
from pyspark.ml.evaluation import ClusteringEvaluator,
MulticlassClassificationEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml.stat import Correlation
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os


# --- START SPARK ---
spark = SparkSession.builder \
    .appName("NASA_Meteorite_Project") \
    .master("local[*]") \
    .config("spark.driver.memory", "4g") \
    .config("spark.sql.execution.arrow.pyspark.enabled", "true") \
    .getOrCreate()

spark.sparkContext.setLogLevel("WARN")
print(f"Spark Session Running. Version: {spark.version}")


# --- LOADING & CLEANING ---
print("--- DATA PROCESSING ---")

# 1. Load Data
try:
    df = spark.read.option("header", "true") \
                   .option("inferSchema", "true") \
                   .csv("meteorite-landings.csv")
    print(f"Original Rows: {df.count()}")
except Exception as e:
    print("ERROR: 'Meteorite_Landings.csv' not found.")
    raise e
```

```
# 2. Advanced Cleaning (Removing Nulls & 'Null Island' coordinates)
df_clean = df.na.drop(subset=["mass", "reclat", "reclong", "year", "fall"])
\
              .filter((col("mass") > 0) &
                      (col("reclat") != 0.0) &
                      (col("reclong") != 0.0))

print(f"Cleaned Rows:  {df_clean.count()}")
df_clean.show(5)


# --- FEATURE ENGINEERING ---
print("--- FEATURE ENGINEERING ---")

# A. Log Transformation (Normalize mass)
df_prepared = df_clean.withColumn("log_mass", log1p(col("mass")))

# B. Index Target (Fell/Found -> 0/1)
indexer = StringIndexer(inputCol="fall", outputCol="label")
df_indexed = indexer.fit(df_prepared).transform(df_prepared)

# C. Vector Assembly
feature_cols = ["reclat", "reclong", "log_mass", "year"]
assembler = VectorAssembler(inputCols=feature_cols,
outputCol="raw_features")
df_assembled = assembler.transform(df_indexed)

# D. Scaling (Standardization)
scaler = StandardScaler(inputCol="raw_features", outputCol="features",
                        withStd=True, withMean=True)
scaler_model = scaler.fit(df_assembled)
df_final = scaler_model.transform(df_assembled)

# Split Data
train_data, test_data = df_final.randomSplit([0.8, 0.2], seed=42)
print(f"Training Set: {train_data.count()} | Testing Set:
{test_data.count()}")


# --- CORRELATION MATRIX ---
print("--- GENERATING CORRELATION MATRIX ---")

# Calculate Correlation
vector_col = "corr_features"
assembler_corr = VectorAssembler(inputCols=feature_cols + ["label"],
outputCol=vector_col)
df_corr = assembler_corr.transform(df_indexed)
matrix = Correlation.corr(df_corr, vector_col).head()
corr_array = matrix[0].toArray()

# Plot Heatmap
plt.figure(figsize=(10, 8))
labels = feature_cols + ["label"]
sns.heatmap(corr_array, annot=True, fmt=".2f", cmap='coolwarm',
            xticklabels=labels, yticklabels=labels)
plt.title("Feature Correlation Matrix")
plt.show()
```

```
# --- K-MEANS CLUSTERING ---
print("--- UNSUPERVISED LEARNING ---")

kmeans = KMeans(featuresCol="features", k=5, seed=1)
model_km = kmeans.fit(train_data)
predictions_km = model_km.transform(test_data)

evaluator_km = ClusteringEvaluator()
silhouette = evaluator_km.evaluate(predictions_km)
print(f"K-Means Silhouette Score (k=5): {silhouette:.4f}")


# --- MODEL COMPARISON ---
print("--- SUPERVISED LEARNING COMPARISON ---")

def evaluate(preds, name):
    acc =
MulticlassClassificationEvaluator(metricName="accuracy").evaluate(preds)
    print(f"{name} Accuracy: {acc:.4f}")
    return acc

# 1. Logistic Regression
lr = LogisticRegression(featuresCol="features", labelCol="label",
maxIter=20)
evaluate(lr.fit(train_data).transform(test_data), "Logistic Regression")

# 2. Decision Tree
dt = DecisionTreeClassifier(featuresCol="features", labelCol="label")
evaluate(dt.fit(train_data).transform(test_data), "Decision Tree")

# 3. Gradient Boosted Trees
gbt = GBTClassifier(featuresCol="features", labelCol="label", maxIter=20)
evaluate(gbt.fit(train_data).transform(test_data), "Gradient Boosted
Trees")


# --- CROSS-VALIDATION ---
print("--- TUNING RANDOM FOREST ---")
rf = RandomForestClassifier(featuresCol="features", labelCol="label")

# Grid Search Configuration
paramGrid = ParamGridBuilder() \
    .addGrid(rf.numTrees, [20, 50]) \
    .addGrid(rf.maxDepth, [5, 10]) \
    .build()

# 3-Fold Cross Validation
crossval = CrossValidator(estimator=rf,
                          estimatorParamMaps=paramGrid,

evaluator=MulticlassClassificationEvaluator(metricName="accuracy"),
                          numFolds=3)

print("Training Model.....")
cv_model = crossval.fit(train_data)
best_rf_preds = cv_model.transform(test_data)
```

```
print("Best Random Forest Results:")
evaluate(best_rf_preds, "Random Forest (Tuned)")


# --- EXPORT FOR TABLEAU ---
print("--- EXPORTING RESULTS ---")

# 1. Get Supervised Predictions & Rename to avoid conflict
final_predictions = cv_model.bestModel.transform(df_final)
final_predictions = final_predictions.withColumnRenamed("prediction",
"RF_Prediction")

# 2. Get Clustering Labels
final_export = model_km.transform(final_predictions)
final_export = final_export.withColumnRenamed("prediction", "Cluster_ID") \
                           .withColumnRenamed("label", "Actual_Label")

# 3. Select Columns for Visualization
output_df = final_export.select(
    "name", "recclass", "mass", "year", "reclat", "reclong",
    "Actual_Label", "Cluster_ID", "RF_Prediction"
)

# 4. Save to CSV
csv_name = "Meteorite_Tableau_Results.csv"
try:
    # Using Pandas for safe local saving
    output_df.toPandas().to_csv(csv_name, index=False)
    print(f"SUCCESS! File saved as: {os.path.abspath(csv_name)}")
    print("Action: file is for Tableau for visualization.")
except Exception as e:
    print("Error exporting:", e)
```

Appendix D: Dataset Link

Kaggle : https://www.kaggle.com/datasets/nasa/meteorite-landings