

Babu Banarasi Das University

BBD City, Faizabad Road, Lucknow Uttar Pradesh



PROJECT - AutoInsurance Fraud DETECTION Using IBM SPSS Modeler

**SUBMITTED TO:
Mr. AYUSHMAN
BHADAURIA**

**SUBMITTED BY:
VISHWAJIT VISHWAS**

Auto Insurance Fraud Detection Using C&R Tree Algorithm

Agenda / Definition

The project aims to detect fraudulent insurance claims using the C&R Tree (Classification and Regression Tree) method in IBM SPSS Modeler.

By analyzing claim data (such as vehicle details, claim amount, and customer info), the model identifies patterns and predicts whether a claim is fraudulent (Y) or non-fraudulent (N).

Outcomes / Learning

- Import and explore a dataset in IBM SPSS Modeler
- Perform data cleaning (remove irrelevant columns, handle missing values)
- Partition data into training and testing samples
- Build and evaluate a C&R Tree classification model
- Generate and interpret prediction results and graphs

This project demonstrates the full data mining workflow — from preparation to model evaluation.

Required Tools

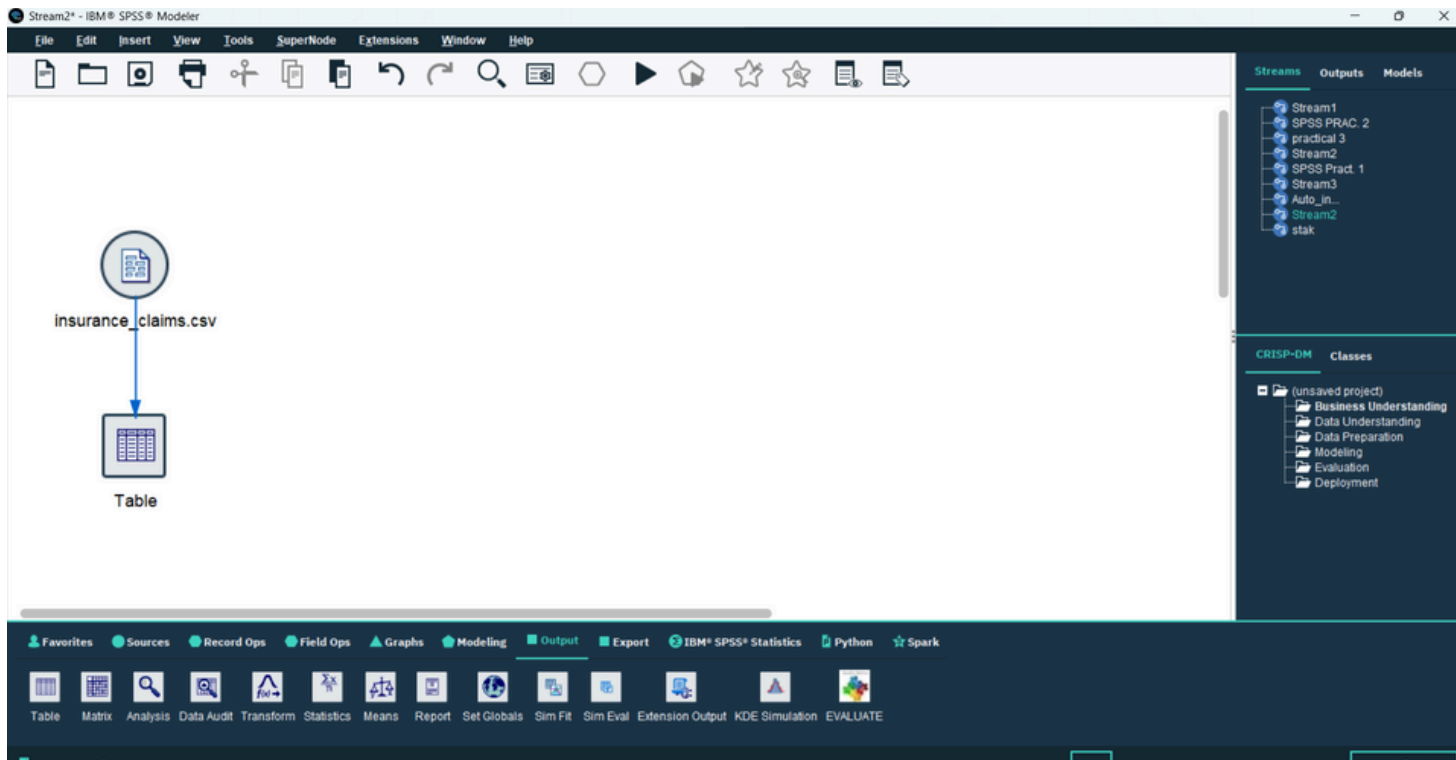
The tool used for this project is IBM SPSS Modeler.

Working

- **The project involves:**
- Importing the insurance claim dataset
- Cleaning and preparing the data
- Setting variable roles and partitioning data
- Configuring and running the C&R Tree model
- Viewing prediction results in a table and histogram
-

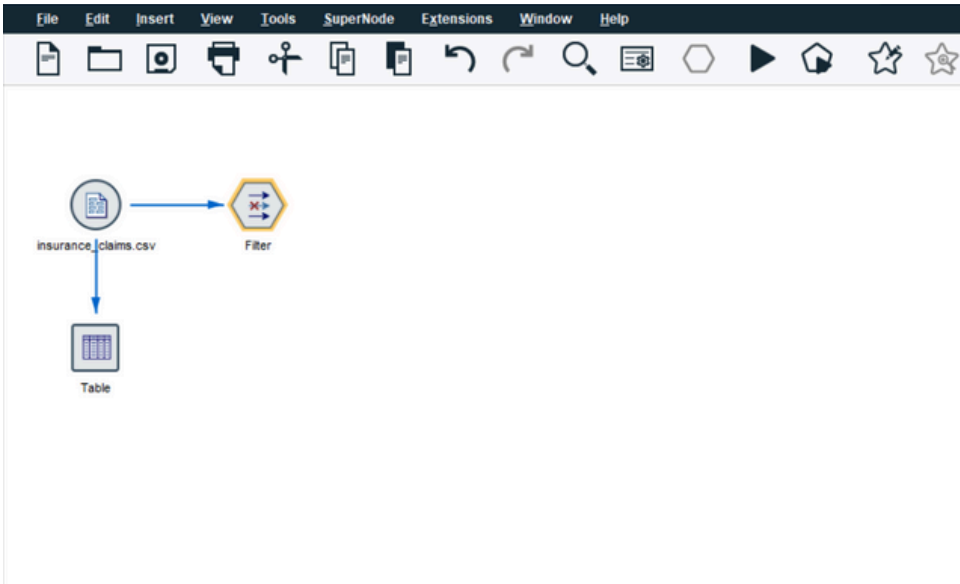
Step 1: Import Data

Loaded the dataset (insurance_claims.csv) into SPSS Modeler using the Var.File Node under Sources palette After reading metadata, all fields were correctly recognized.



Step 2: Remove unnecessary Data

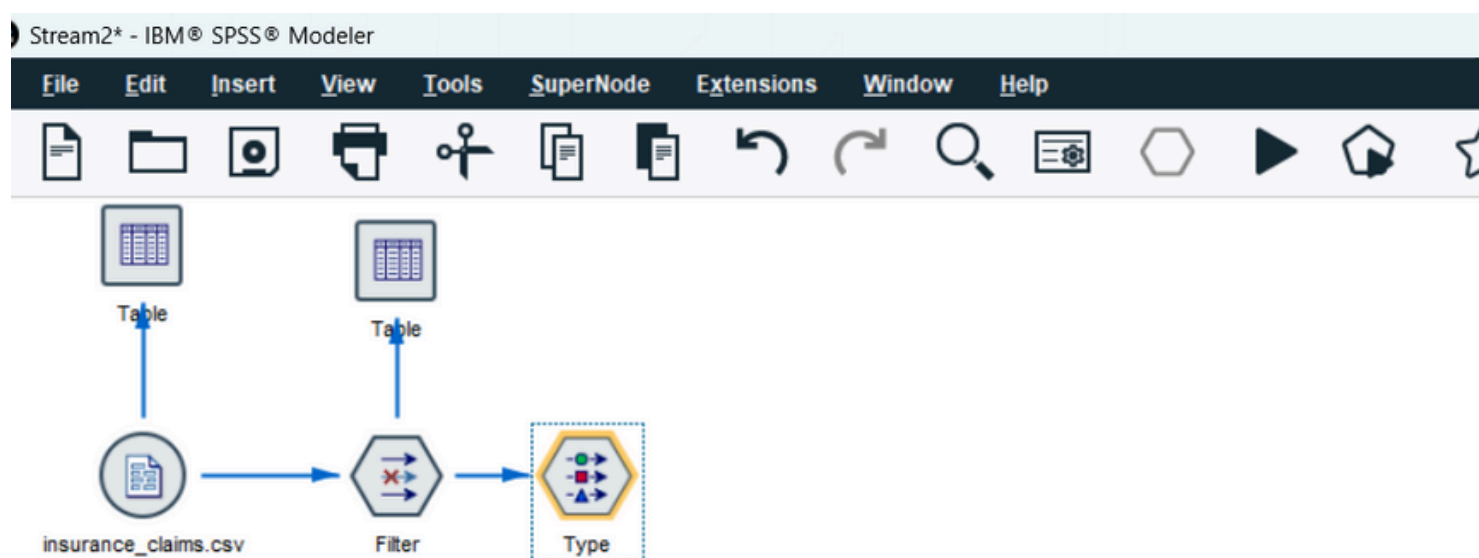
The Filter Node was used to exclude the irrelevant column _c39 from the dataset. This column contained empty or meaningless values that could interfere with model accuracy. By filtering it out, we ensured that only useful fields (such as claim amount, vehicle claim, auto make, model, year, and fraud status) were retained for analysis.



Step 4 : Type Node

Defined roles for each field:

- Input Fields: Claim amount, incident severity, age, etc.
- Target Field: fraud_reported



Preview

Types Format Annotations

Read Values Clear Values Clear All Values

| Field | Measurement | Values | Missing | Check | Role |
|------------------|-------------|--------|---------|-------|--------|
| total_claim... | Continuous | <Read> | | None | Input |
| injury_claim | Continuous | <Read> | | None | Input |
| property_clai... | Continuous | <Read> | | None | Input |
| vehicle_claim | Continuous | <Read> | | None | Input |
| auto_make | Categorical | <Read> | | None | Input |
| auto_model | Categorical | <Read> | | None | Input |
| auto_year | Continuous | <Read> | | None | Input |
| fraud_report... | Categorical | <Read> | | None | Target |

☒ View current fields ☐ View unused field settings

OK Cancel Apply Reset

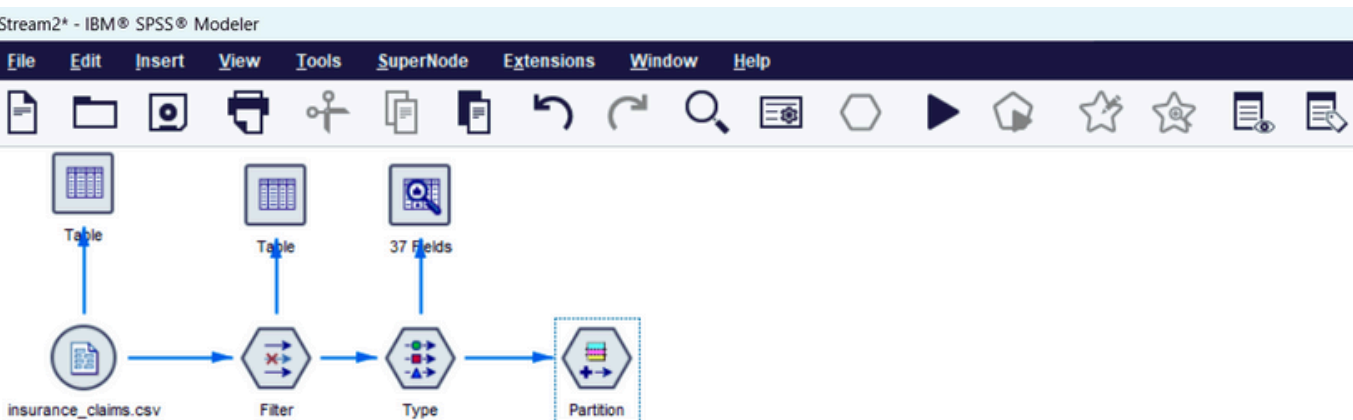
Step 4 : Partition Data

Added Partition Node to split data:

70% for Training

30% for Testing

This allows model evaluation on unseen data



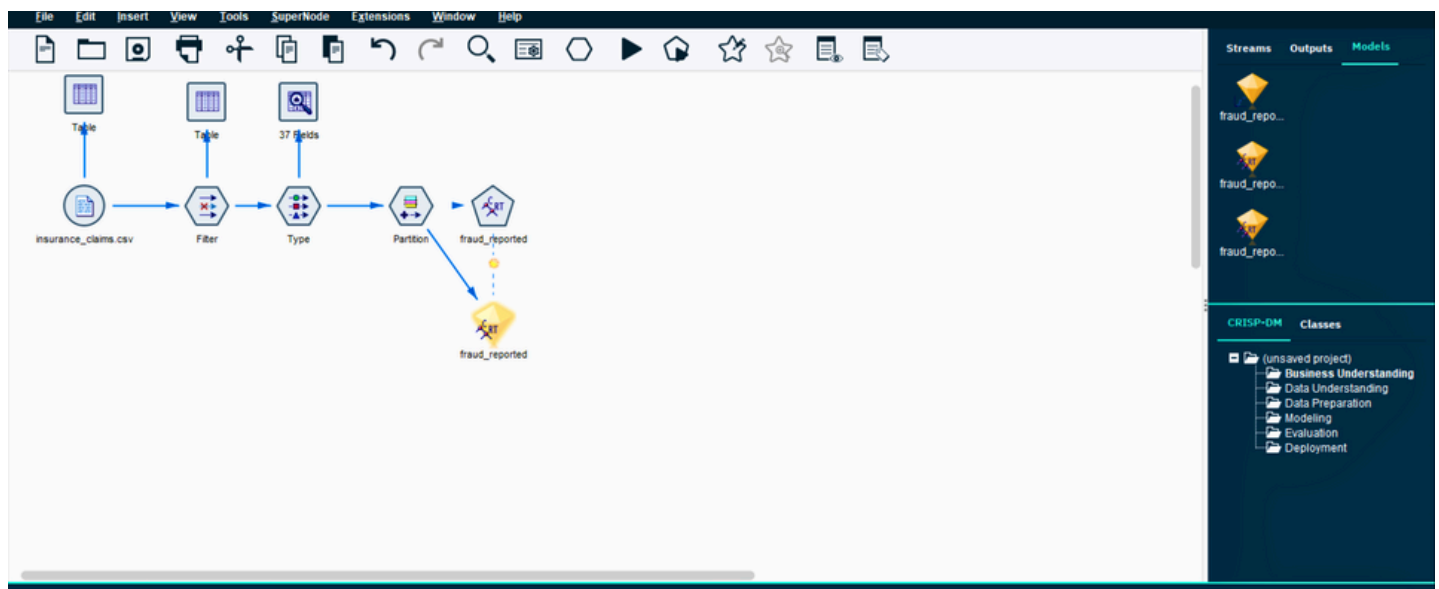
The screenshot shows the configuration dialog for the Partition node. The 'Settings' tab is active. The 'Partition field' is set to 'Partition'. The 'Partitions' section has the 'Train and test' radio button selected. The 'Training partition size' is 70, 'Testing partition size' is 30, and 'Validation partition size' is 0. The 'Total size' is 100%. The 'Values' section has the 'Append labels to system-defined values' radio button selected. The 'Repeatable partition assignment' checkbox is checked. The 'Seed' is 1234567. The 'Generate' button is visible. The 'Use unique field to assign partitions' checkbox is unchecked.

| Partition field: | Partition | | |
|---|--|---|------------------------|
| Partitions: | <input checked="" type="radio"/> Train and test <input type="radio"/> Train, test and validation | | |
| Training partition size: | 70 | Label: Training | Value = "1_Training" |
| Testing partition size: | 30 | Label: Testing | Value = "2_Testing" |
| Validation partition size: | 0 | Label: Validation | Value = "3_Validation" |
| Total size: | 100% | | |
| Values: | <input type="radio"/> Use system-defined values ("1", "2" and "3") | | |
| | <input checked="" type="radio"/> Append labels to system-defined values | | |
| | <input type="radio"/> Use labels as values | | |
| <input checked="" type="checkbox"/> Repeatable partition assignment | | | |
| Seed: | 1234567 | <input type="button" value="Generate"/> | |
| <input type="checkbox"/> Use unique field to assign partitions: | | | |

Step 5 : Build Models

Model - C&R Tree

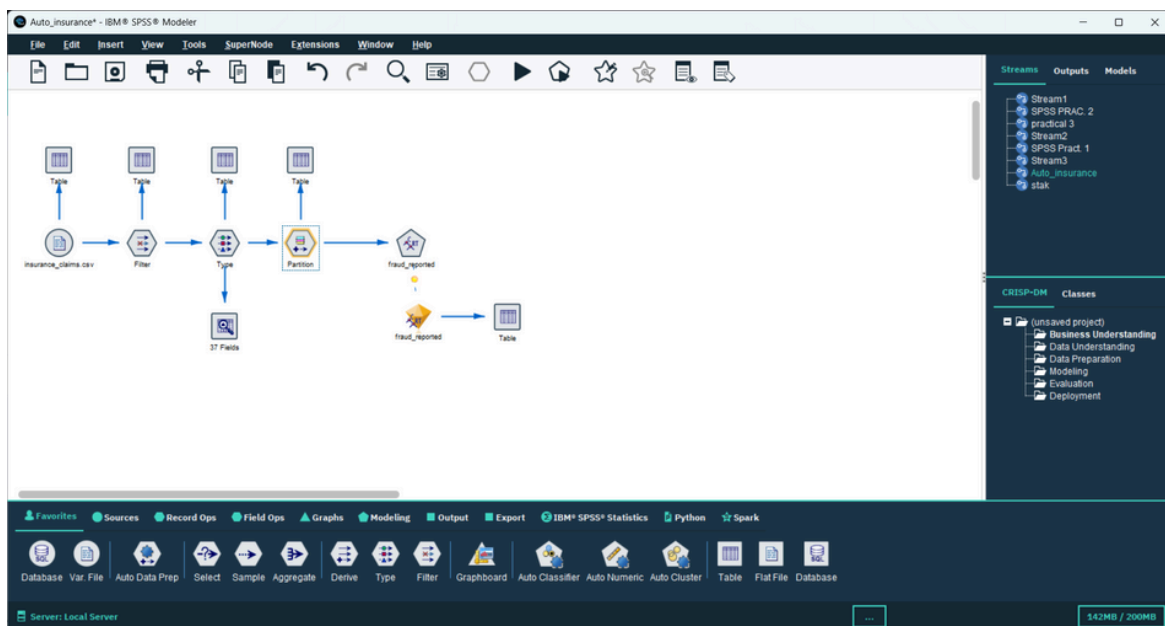
- From the Modeling palette, drag a C&R Tree Node.
- Connect it to the Partition Node.
- Open it → confirm:
- Target: fraud_reported
- Inputs: Auto-selected.
- Click Run → view the decision tree output (splits, accuracy, etc.).



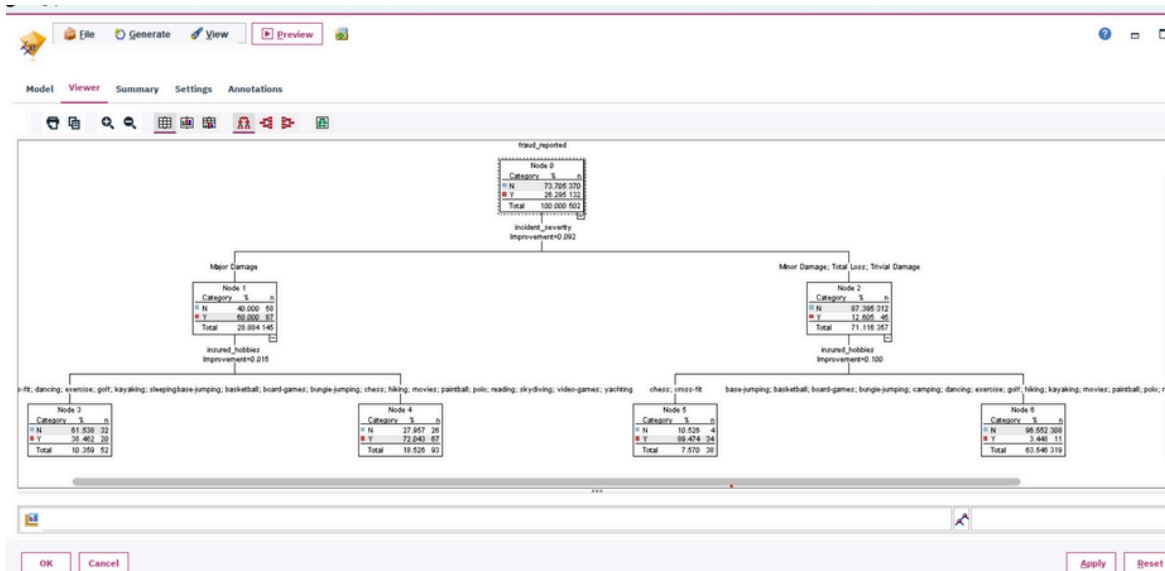
Insights :

- Major Damage and High Claim Amounts are the most common indicators of fraud.
- Customers with risky hobbies (like skydiving or motor racing) are more prone to fraudulent claims.
- Fraud detection is better when combining C&R Tree visualization

FINAL VIEW



OUTPUT:



Conclusion :

The project successfully built and evaluated two models to detect insurance fraud using IBM SPSS Modeler.

The models help the insurance company:

- Identify potential fraudulent claims early.*
- Save costs by reducing false claims.*
- Improve the reliability of claim verification systems.*

Overall, the C&R Tree and Logistic Regression models provided valuable insights into fraudulent behavior patterns.